Издательство «Знание»

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Пользователям бытового компьютера

1990

8

Новое
в жизни,
науке,технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

8/1990

Издается
ежемесячно
с 1988 г.

ПОЛЬЗОВАТЕЛЯМ БЫТОВОГО КОМПЬЮТЕРА

В номере:

Н.М.Саттаров
Состояние программного обеспечения микро-ЭВМ
В.В.Ермаков
Многомерные массивы на БЕЙСИКе

РУБРИКИ

Нам пишут
Как "убить" машинное время
БК за рога

Издательство
"Знание"
Москва
1990

ББК 32.97
П 53

Авторы выпуска

ЖАРИКОВ Леонид Николаевич — научный сотрудник, занимается разработкой программного обеспечения АСУ.

УСЕНКОВ Дмитрий Юрьевич — студент МГТУ им.Баумана, программист.

ЛОНГИНОВ Александр Сергеевич — журналист. Основная направленность публикаций — пути развития научно-технического прогресса.

В этом номере приняли участие наши читатели:

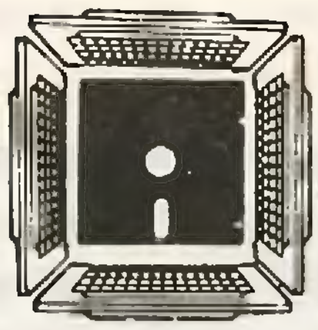
БЕРЕЗЕНЦЕВ И.П., ГАРМАШОВ А.П., САТТАРОВ Н.М., БЕНДАРИНОВ В.Г., ЕРМАКОВ В.В., КОНЮШЕНКО А.М., САПОГОВ В.В., МЕЛЕНТЬЕВ А.В., СИНЯГИН С.Ю.

РЕЦЕНЗЕНТ — *Л.Н.Жариков*

РЕДАКТОР Б.М.ВАСИЛЬЕВ

ISBN 5-07-001511-7

© Издательство "Знание", 1990 г.



Этот выпуск предназначен для пользователей БК 0010 и, соответственно, читателей рубрики "БК за рога". Весь текст составлен по письмам наших читателей, обсуждающих очень интересный вопрос — быть или не быть БК 0010, и не менее интересный — как работать в условиях почти полного отсутствия периферии к БК 0010.

Я — ВЛАДЕЛЕЦ БК 0010.01

В печати не перестают появляться заметки и обширные статьи о том, что БК — детская игрушка и его пора снимать с производства...

Мне надоело читать и слушать насмешки о доступных пользователю 16 кБ ОЗУ и досужие рассуждения о преимуществах процессоров Ай-Би-Эм. В конце концов!!! Процессор 1801ВМ1 — ничуть не хуже. Разработаны схемы расширения "+ 32","+ 128" и "+ 256", не требующие "потрошения" внутренностей БК при сохранении работоспособности всего уже имеющегося программного обеспечения без доработок. Расширение предоставляет пользователю адресное пространство до 4 МБ при организации быстрого обмена данными с внешними устройствами как с ячейками памяти. Расширение позволяет создавать мощное программное обеспечение, сравнимое и превосходящее возможности "фирмы", не содержит "экзотических" микросхем и доступно для повторения радиолюбителю средней квалификации. Однако, не являясь профессиональным системным программистом, испытываю затруднения в разработке "зрелого" програмного обеспечения расширения. Прошу откликнуться добровольцев, желающих принять участие в разработке операционной системы под "БК +".

Анатолий Михайлович Конюшенко
220030, Минск-30, а/я 1.

Н.М. Саттаров
(г.Казань, тел. 38-34-03)

СОСТОЯНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРО-ЭВМ

Необходимость простых моделей компьютеров

Наряду с мощной вычислительной техникой типа VAX, IBM, PC/XT/AT, PS/2 существуют самые простые модели компьютеров: БК0010, МИКРОША, ЛЬВОВ и другие. Попробуем определить сферу применения микро-ЭВМ. Каковы их возможности? Что это: настольные калькуляторы или полноправные компьютеры? Здесь чувствуется непонимание даже у серьезных специалистов.

Можно услышать мнение о необходимости прекратить выпуск таких компьютеров. Один академик выскалзся, что даже знакомство с ними вредно. Ему почему-то хочется повсеместно внедрить систему UNIX (?). Многие ратуют за полную ориентацию страны на IBM PC, PS/2.

Это прекрасные замыслы. Но не будем мечтать. Подобных машин насчитываются единицы даже в "программистских" учреждениях, где на 40 профессионалов — один "персональный" компьютер. А что делать простым инженерам, студентам, школьникам и просто любителям? Ждать эры всеобщей компьютеризации?

Будем реалистами. Мощная техника пока не доступна массовому пользователю. Прежде всего из-за высокой стоимости в десятки тысяч рублей. Нам нужны простые и дешевые компьютеры. Кстати, на Западе такие модели до сих пор существуют и пользуются спросом.

Сторонники грандиозных идей не хотят понять, что со многими пробле-

мами вполне могут справиться простые микро-ЭВМ, что для ряда производственных задач использование дорогой техники просто нерентабельно. Мы не так богаты, чтобы стрелять из пушек по воробьям.

У каждого класса компьютеров — своя область применения. Большие машины — для больших проблем, малые — для проблем поменьше, но для тысяч и тысяч пользователей. Микро-ЭВМ подобны своим старшим братьям по архитектуре и принципам программирования. Отличие — в объеме памяти, быстродействии, а следовательно, в сложности и размерности решаемых задач.

БК0010 — самый популярный микрокомпьютер

О причинах популярности БК0010 было много сказано. Небольшая стоимость означает возможность (и безопасность) приобрести его для дома. Высокая надежность даже немного странна для представителя отечественной техники. Система команд 16-разрядная — программировать в машинных кодах проще, чем в 8-разрядных системах. Совместимость команд с СМ-4, ДВК, Электроника-60 позволяет разрабатывать на этих машинах программы для БК. Большому количеству специалистов архитектура и команды БК0010 знакомы и привычны, не надо переучиваться. Графические возможности компьютера позволяют создавать рисунки на экране дисплея (этим не балует, например, Микроша). Быстродействия вполне достаточно для создания динамических (движущихся) рисунков.

Таким образом, много положительного уже заложено. Но ЭВМ без программного обеспечения — просто соединение металла, пластмассы и проводочков. Степень полезности компьютера определяет количество и качество его программ. Что же у нас есть?

Один восторженный журналист недавно написал, что, оказывается, для БК0010 имеется не 10 программ, поставляемых заводом-изготовителем, а целых 300. Все-таки заметим, что уже в 1988 году в некоторых архивах имелось до 900 программ. Сколько сейчас — трудно сказать. Да и нужно ли считать, сколько всего поштучно: гвоздей, булавок и пуговиц. Важно другое: много ли красивых и сложных программ, есть ли самые необходимые?

Да, у БК0010 имеются самые нужные системные программы: трансляторы ассемблера, отладчики, копировщики, текстовые и графические редакторы, специальные системы и языки программирования. Микро-ЭВМ используется с различными устройствами: принтерами, "мышью", джойстиком, координатными планшетами, графопостроителем. Создано много обучающих программ, целые пакеты программ по предметам школьных и вузовских курсов. И, конечно, наработано большое количество игровых и занимательных программ.

Эволюция ассемблеров БК0010

Интерпретаторов ФОКАЛа и БЕЙСИКа для создания программной одежды БК0010 явно недостаточно.

Известно, что самые красивые и эффективные программы разрабатываются в машинных кодах. Инструментом служат специальные программы — ассемблеры. Это главнейший инструмент, без которого вообще нельзя сделать ничего серьезного.

Сразу же потребовалось средство создания программ для БК силами самой БК. Увы, разработчики БК0010 "забыли" приложить ассемблер. Будь он прошит в ПЗУ вместо никому не нужных тестов самоконтроля и директив отладки, уровень программного обеспечения микро-ЭВМ был бы совсем иным. Сколько идей не реализовано только из-за отсутствия хорошего инструмента. Сколько лишних сил потрачено при использовании изуверских методов программирования.

Первопроходцам БК0010 при-

шлось начинать с нуля. Нашлись энтузиасты, которые либо на других ЭВМ, либо на ФОКАЛе, а то и в числовых кодах создали простейшие отладчики-ассемблеры. На их основе разрабатываются более сложные: GROT, ОТЛ9, OS0010. Идет развитие на основе собственных предыдущих версий. Иголка с ниткой совершенствуется — и БК0010 понемногу одевается: появляются интересные программы в машинных кодах.

Работать в ассемблерах-отладчиках сложно: нужно помнить номера ячеек памяти и их назначение, трудно делать бесконечные исправления. Это любительский метод работы. Риск сойти с ума растет пропорционально объему создаваемой программы. Все-таки назначение отладчиков — отладка или исправление чужих программ.

У собственной программы должен быть исходный текст: с метками, комментариями. Все исправления делаются только в тексте, а машинные коды создаются каждый раз заново при трансляции.

Начинают появляться первые ассемблеры-трансляторы. Сначала робкие: без встроенных редакторов текста (например, программа ТР 5Е), без аппарата локальных меток (МИКРО2). Затем разрабатываются хорошие трансляторы МИКРО8, МИКРО10 (С.Шмытов). Но хочется еще лучше. Нужны встроенные средства отладки, нужно использовать все ресурсы памяти. МИКРО8 дорабатывается автором этой статьи в МИКРО8+. В базовую программу вставляется мини-операционная система и более совершенный редактор текста. Но ресурсы памяти еще есть: можно плотнее упаковать исходный текст. Автором разрабатывается новый транслятор: ASM. Это результат эволюции ассемблеров БК0010 к 1990 г.

Размер системы ASM — 4 кбайта. Основное преимущество нового инструмента — существенное увеличение длины транслируемых в памяти программ (до 5-10 кбайт).

Система состоит из монитора, редактора текста и компилятора. Монитор выполняет функции мини-операционной системы: чтение — запись на магнитную ленту исходного текста и скомпилированных модулей. Из монитора вызываются редактор текста, компилятор и созданная машинная программа. Предусмотрены удобные средства отладки: просмотр или изменение ячеек памяти в форматах слова, байта, символа. Монитор перехватывает несанкционированные прерывания отлаживаемых программ.

Для создания текста на языке Ассемблера используется встроенный редактор TED6. Текст упаковывается в 2,5 раза плотнее, чем в системе МИКРО8. Поддерживается работа в режимах обычной и расширенной памяти, 32 и 64 символа в строке.

Компилятор трехпроходный, но по скорости превосходит однопроходный МИКРО8. Реализован аппарат внутренних и локальных меток. Имеются псевдокоманды записи слов, байтов, текста; вывода регистров; включения-выключения компиляции.

Печать текста на принтер (два вида интерфейса) и перекодировка текста из формата МИКРО8 осуществляются вспомогательной программой T-ASM.

Ниже приведен пример программы на языке Ассемблера.

```
     MOV # PRI, R1
     CLR R2
     EMT 20
     .REG                печать регистров
     RTS PC
PRI: .B:14,221,234       очистка, цвет, инверсия
     .A:ПРИМЕР
     .E                  запись нуля
```

Программы работы с файлами

Для перезаписи файлов с одной кассеты на другую потребовались специальные программы — копировщики. Принцип их работы следующий: исходный файл сначала прочитывается в память микро-ЭВМ с магнитной ленты, а затем записывается на другую ленту. Несмотря на кажу-

щуюся простоту, реализация подобной программы достаточно сложная.

Любителями БК созданы хорошие копировщики COPY 4, COPIR (С.Ивашинников). Однако простого копирования при работе с файлами часто недостаточно. Автором статьи разрабатываются программы HELP и HELP3, где реализованы следующие функции операционных систем: DIR, COPY, RENAME, VERIFY, TYPE, PRINT — просмотр оглавления МЛ, копирование, изменение имени, контроль файлов на МЛ, просмотр текстовых сообщений, печать текста на принтер.

Можно просмотреть или изменить файл в форматах слова, байта, текста. В программу HELP3 встроен простейший редактор текста.

Сравнительные характеристики программ работы с файлами приведены в таблице.

Таблица

Функции	COPY4	COPIR	HELP	HELP3
Максимальная длина файла	70000	37000	70000	40000
Количество скоростей записи	1	3	1	4
Чтение оглавления ленты	да	-	да	да
Чтение в экранную область	-	да	да	да
Изменение имени файла	да	-	да	да
Изменение длины файла	да	-	да	да
Изменение адреса загрузки	да	-	да	да
Запись файла потоком	да	да	да	да
Запись файла блоками	-	-	да	да
Запуск файла на выполнение	да	-	да	да
Контроль файлов	да	-	да	да
Просмотр файла	-	-	да	да
Изменение файла	-	-	да	да
Печать на принтер	-	-	да	да
Интерактивное меню	-	-	-	да

Важнейшей функцией является возможность записи файлов по блокам. Длина блока задается произвольно. Такие файлы снабжаются автозагрузчиком и читаются из пускового монитора БК0010 обычным образом. Порча какого-либо блока не приводит к порче всего файла: эти блоки можно прочитать с другого участка ленты. Введение блочных файлов позволило значительно повысить надежность хранения информации.

Редакторы текста

Современный человек постоянно что-то пишет: деловые документы, письма, статьи. Текстовые редакторы нужны всем. Это самые популярные и используемые программы во всем мире.

Попробуем определить минимальный набор функций, необходимых для редактирования текста. Это исправления внутри строки (замена символов, сдвижка, раздвижка); вставка и удаление строк; листание текста (вверх, вниз, в начало, в конец). При держании клавиши в нажатом состоянии можно имитировать повторное нажатие клавиши. Перечисленные функции редакторов — стандарт и должны присутствовать всегда.

Для БК0010 существует множество текстовых редакторов. Автору известны следующие: TR, TRED, EDIT, REDTT1, EDASP5, EDASP10, редакторы систем МИКРО10, FORTHV83.

В некоторых программах реализованы сложные и реже используемые функции: карманы, поиск фрагментов и другие. Но увы, минимальный набор команд нигде не присутствует полностью. Общая тенденция — управление редактированием "посажено" на те клавиши, которые больше понравились разработчикам. Но есть ведь определенные стандарты. Во всех перечисленных редакторах, кроме МИКРО10, не осуществляется упаковка пробелов, а это слишком роскошное использование маленькой памяти БК.

Автором разработан текстовый редактор TED6. Эта программа обладает наибольшей емкостью текста: в памяти микрокомпьютера можно создавать 5 — 11 листов текста формата А4 (60х60 символов). Кроме минимального набора функций редактирования, реализована работа с карманами и выравнивание текста по правому краю листа. Предусмотрена возможность записи текста на МЛ в виде автономного программного блока. Печать на принтер осуществляется для двух видов интерфейса. Строчные буквы могут быть преобразованы в заглавные. Есть возможность постраничной распечатки текста. Редактор имеет встроенное описание — это удобно для начинающих. Допустима работа в режимах 32 и 64 символа в строке, обычной и расширенной памяти БК0010.

TED6 — хорошее средство для создания текстовых документов. Данная статья создана при помощи этого редактора.

Графические редакторы

Простейшие возможности рисования на экране дисплея защиты в ПЗУ БК0010. Попытки создания графических редакторов предпринимались, но до последнего времени не было ничего серьезного. Это связано со сложностью разработки.

Недавно в Москве появился мощный редактор, позволяющий при помощи "мышки" рисовать на экране, записывать рисунок на ленту, распечатывать его на принтер. Имеется несколько видов шрифта, реализованы функции окна, резиновой нити. Редактор имеет удобное меню.

Автором разработан редактор GRAF1. Здесь заложен принцип векторного хранения информации (запоминаются координаты опорных точек). Основные графические примитивы — прямая, окружность, дуга, текст (6 видов шрифта). Опорные точки примитивов задаются положением графического курсора (пера). Перо управляется при помощи клавиатуры, "мышки", джойстика, координатного планшета. Рисунок можно вывести на графопостроитель (СМ6470) или принтер (допустима загрузка любого драйвера).

Каждый рисунок имеет имя и хранится в графической библиотеке. Разрешается создание 45 рисунков в памяти БК общим объемом до 1500 примитивов. Рисунки можно масштабировать, поворачивать, вкладывать друг в друга (5 уровней вложенности). Предусмотрено чтение или запись библиотеки на магнитную ленту. Можно создать, удалить, исправить рисунок.

Работа осуществляется в интерактивном режиме. Главное меню напоминает известный NORTON COMMANDER для IBM PC. Установка директив меню, поиск рисунков производится наглядным инверсным указателем. В программу встроено краткое описание.

Графическим редактором GRAF1 можно создавать компьютерные фильмы; имеются средства простейшей динамики. Предполагается развитие редактора в направлении усиления динамики.

Обучающие и демонстрационные программы

Компьютерное обучение, широко используемое во всем мире, пришло и в нашу страну. Появление недорогих компьютеров сделало обучающие программы доступными учащимся, а не только проверяющим комиссиям.

Применение автоматизированного обучения повышает скорость и каче-

ство усваивания материала. Наглядность графических построений, динамика, музыка делают обучение понятным и интересным.

Много обучающих программ для БК0010 создано на интерпретаторах ФОКАЛа и БЕЙСИКа. Однако достижение хороших динамических эффектов ограничивается медленностью интерпретаторов. При создании программ на Ассемблере нужное быстродействие достигается, но значительно увеличивается трудоемкость.

Работа на языках программирования общего назначения достаточно сложна и требует определенной квалификации специалиста. В таких условиях трудно поставить процесс создания обучающих программ "на конвейер". Поэтому предпринимаются попытки систематизировать демонстрационные эффекты, собрать их воедино и предоставить разработчику в специальном инструменте. Такой подход значительно облегчает дело.

Для создания обучающих программ и игр, содержащих динамические иллюстрации, звуковые эффекты, разработан Т-язык (Ю. Кузьмин).

Автором статьи разработана система создания демонстрационных программ SYST4. Основные ее возможности следующие: создание статических и динамических рисунков, различные способы вывода текста, музыкальные эффекты, контроль знаний.

Программы составляются на специальном языке демонстрационных программ, содержащем 36 различных операторов. Исходный текст создается при помощи встроенного экранного редактора. Выполнение программы происходит в интерпретирующам режиме. Динамические рисунки обладают хорошим быстродействием; вывод графики реализован в 2.5 раза быстрее, чем в системных программах ПЗУ.

Ниже приводится пример использования языка демонстрационных программ.

```
1:MUSC   140,125,110      ; музыка: до,
                            ре, ми
2:LINE   10,10,100,10,1,0 ; движущаяся
                            линия
  DRAW   BM 10,10 M10,100
         R10 A100,100,10  ; рисунок
  TXTL   16
         вывод текста с
         16-й строки $
  CALL   1,2               ; вызов операторов 1,2
```

Система достаточно компактна (4,5 кбайта) и позволяет создавать значительные по объему фильмы (в 3-4 раза длиннее, чем на БЕЙСИКе БК0010.01). Здесь заложен целый комплекс демонстрационных эффектов. Программисту не надо задумываться, например, о том, как создать движущийся рисунок. Решается конкретная задача: какой рисунок и куда переместить, какое выбрать музыкальное сопровождение.

Трудоемкость создания демонстрационных фильмов в рамках системы существенно ниже, нежели при использовании универсальных языков программирования. Работа не требует знания сложных приемов и вполне под силу начинающему.

Игровые программы

Эта группа программ наиболее многочисленная. Здесь и шахматы (С.Шмытов), и шашки, и интересные динамические игры. Наверное, не стоит все перечислять и описывать. Только отметим, что некоторые игры БК красивее своих прародителей на IBM PC. Например, RUNNER (А.Марков), TETRIS (А.Л.Пажитнов). Это говорит о зависимости красоты программы не столько от процессора компьютера, сколько от таланта программиста.

Перспективы развития БК0010

Выпущенные модели микро-ЭВМ уже нашли своего пользователя. Созданы десятки клубов любителей

БК0010. Этот компьютер не залеживается на прилавках магазинов.

Но перспективы определяют заводы-изготовители. И порадовать им нечем. В каждой модели БК заложена какая-нибудь странность. БК0010.01 удивила "новой" клавиатурой: ненадежной и дребезжащей; к тому же клавиш стало на десяток меньше. Опытные образцы БК0011 имеют подобную клавиатуру; огорчает грустный дизайн дисководов и бедность поставляемых программ. Разработчики обнадеживают: мол, в скором времени будут и текстовые, и графические редакторы, и многомного всего. Ничего нет и совместимости с БК0010 тоже нет. Но сколько раз можно начинать с нуля? И, наконец, цена "бытового компьютера" БК0011 стала совсем не бытовая.

По дошедшим слухам, один завод мечтает вместо БК выпускать PC-образную ЭВМ, другой носится с идеей создать машину в архитектуре PDP аж с винчестером. Интересно, сколько такое будет стоить? Упрямая черта нашей экономики: стремиться делать подороже и весом поболее.

БК0010 завоевал популярность по причине малой стоимости. Он удачно заполнил нишу особо малых компьютеров. Именно в рамках этой ниши нужно развитие. Но с 1985 г. в БК ничего не изменилось даже в комплекте прилагаемых программ.

Неужели и дальше заводы-изготовители будут пребывать в глубокой спячке? Ведь потребность в таких микро-ЭВМ большая.

В новой модели БК следует увеличить память и быстродействие при условии совместимости с БК0010. Необходим дисковод, причем только один для простоты. И должна быть возможность приобрести новую микро-ЭВМ без дисковода, работающую с магнитной лентой. Нужны две модификации: одна для пользователей и другая, более мощная, для разработчиков программ. Если стоимость окажется невысокой, то новый компьютер будет иметь успех.

В.В.Ермаков
(Магаданская обл.)

МНОГОМЕРНЫЕ МАССИВЫ НА БЕЙСИКЕ

Одним из недостатков вильнюсской версии БЕЙСИКа (24.07.86 г.) является невозможность использования в программах многомерных массивов (Наука и жизнь. — 1989. — № 3; Информатика и образование. — 1989 — № 1). Причина тому — ошибка, внесенная в процессе разработки языка. Суть ее в следующем.

Известно, что адрес A i_1, i_2....i_n элемента массива, например B (N_1, N_2, N_3,....N_n), можно найти, пользуясь соотношением:

$$A = i_1 + i_2.N_1 + i_3N_1N_2 + ... + i_nN_1N_2 N_3....N_{n-1}, \quad (1)$$

где $i_n = 0... N_{n-1}$,
$n = 0... 255$.

В существующей версии языка реализована формула

$$A = i_n + i_2N_1 + i_3N_2 + ... + i_nN_{n-1} \quad (2)$$

Сравнение (1) и (2) показывает, что компьютер верно работает лишь с одно- и двумерными массивами.

Ошибку можно исправить подпрограммой, в которой реализован алгоритм (1):

```
10 ДАТА &013400, &012403, &012601, &05720,
&05303, &03421, &010305, &0166405, &0177776,
&05405, &012602, &010046, &010146, &014001,
&010546, &04737, &0160442, &010102, &012605,
&077507, &062601, &012600, &0754, &0137,
&0160172
20 FOR I% = &0400 TO &0460 STEP & 02
30 READ J%
40 S = S + J%
50 PORE I%, J%
60 NEXT
70 IFS > < 111935 THEN PRINT STRING$ (20,7)
"Ошибка, проверьте данные в строке 10"
80 PRINT "Подпрограмма загружена"
90 END
```

Рис.1

Для этого нужно "заставить" компьютер при вычислении адресов элементов массивов обращаться не к

ошибочной подпрограмме, а к исправной, которая после запуска программы рис.1 будет загружена в ОЗУ с адреса 400_8. Сделать это можно, если внести в программу, которая работает с массивами размерностью более 2, небольшое дополнение

Программа

```
FOR I% = PEEK (&2002) TO PEEK (&02004)
STEP &02 IF PEEK (I%) = -8146 THEN POKE I%,
&0400 NEXT
```

Рис.2

Записать его нужно в текст основной программы до первого обращения к многомерному массиву, лучше всего в начало программы. Это "дополнение" заменит код 160056_8 (-8146_{10}) в скомпилированной программе (адрес подпрограммы с ошибкой) на 400_8 (адрес, куда загружена подпрограмма рис.1). Ячейки 2002_8 и 2004_8 содержат соответственно адреса начала и конца скомпилированной программы. Теперь можно работать с массивами размерностью, как указано в описании языка БЕЙСИК, даже 255, если кому-то удастся описать массив такой размерности оператором DIM.

Для удобства работы машинные коды исправленной подпрограммы можно записать на магнитную ленту командой BSAVE "DIM", &0400, &0460 и загружать ее командой BLOAD"DIM каждый раз, когда работаете с многомерными массивами. Однако если позволяет память, программу рис. 1можно целиком поместить в начало основной программы.

Следует иметь в виду, что в исправленной подпрограмме (см.рис.1) с целью экономии памяти исключена диагностика ошибок. Поэтому перед внесением дополнения (см.рис.2) в текст программы, ее следует отладить.

Как «убить» машинное время

Предлагаю читателям короткую игровую программу "Слалом". Цель — управляя "лыжником" (клавиши ←, →) нужно пройти трассу, огибая камни.

```
5 REM ***SLALOM***
6 REM БЕЙСИК БК-0010-01
7 REM Автор — В.Ермаков
10 CLS
20 A$ = CHR$(182)
30 B$ = CHR$(127)
40 C$ = STRING$(8, "*")
50 PRINT AT (3,4) "Длина трассы 200 метров"
60 PRINT "  уровень 1.....5"
70 I$ = INKEY$
80 IF I$ = "" THEN 70 ELSE KK% = VAL (I$)
90 IF KK% > = 1 AND KK% < = 5 THEN 130
100 PRINT STRING$(7,7)
110 PRINT "НАЖИМАЙ КЛАВИШИ ОТ '1' ДО
'5'"
120 GOTO 60
130 CLS
140 I% = 10%
150 J% = 0%
160 KK% = 200$*KK%
170 POKE &0177660, &0100
180 IF INP (&0177716,&0100) = &O0 THEN
A% = PEEK (&0177662)
    ELSE 200
190 IF A% = 8% THEN I% = I%-1% ELSE IF
A% = 25% THEN I% = I% + 1%
200 IF I% < 0% THEN I% = 0%
210 IF I% > 30% THEN I% = 30%
220 IF POINT (I%*8% + 4%,104%) = 1%
    THEN 340
230 J% = J% + 1%
240 PRINT AT (I%,10) A
250 N% = J%\200% + 1%
260 FOR I% = 1 TO N%
270 PRINT AT (RND(K%)* 31,23) B ;
280 NEXT
290 PRINT
300 IF KK% - J% < 0% THEN 370
310 FOR K% = 1 TO KK%-J%
320 NEXT
```

```
330 GOTO 180
340 PRINT STRING$ (20,7)
350 PRINT AT (3,10) "ВЫ ПРОЕХАЛИ"; FIX
(J%*200/KK%); "M"
360 GOTO 380
370 PRINT AT (2,10) C$"ФИНИШ" C$
380 POKE &0177660, 0
390 GOTO 50
```

ИГРЫ — ДЕЛО СЕРЬЕЗНОЕ

Заключение рецензента

Работать с многомерными массивами в вильнюсской версии БЕЙСИКа БК0010.01 в настоящее время действительно невозможно. Практика показывает: "даже трехмерные массивы оказываются транслятору не по зубам" (Наука и жизнь. — 1989 — №3). Конечно, этот недостаток версии не очень существенный: необходимость работы с многомерными массивами возникает достаточно редко, а организовать использование одномерного массива как многомерного нетрудно.

Можно ввести вспомогательные индексы $i_1, i_2, \ldots i_n$ и рассчитывать каждый раз в подпрограмме индекс одномерного массива "i" по формуле

$$i = i_1 + i_2 N_1 + i_3 N_1 N_2 + \ldots + i_n N_1 N_2 \ldots N_{n-1},$$

где $N_1, N_2, \ldots N_{n-1}$ — размерности воображаемого n-мерного массива; $i_к = 0, \ldots, N_{к-1}$.

При этом должно выполняться условие $N = N_1 N_2 N_3 \ldots N_n$, где N — размерность одномерного массива.

Ясно, что гораздо удобнее пользоваться многомерными массивами в привычной форме. Читатель В.В.Ермаков предложил короткую программу в машинных кодах, позволяющую устранить описанный выше недостаток Бейсика БК0010.01.

Эта подпрограмма в начале сеанса работы загружается в системную область с восьмеричного адреса 400. Затем в начало создаваемой программы на БЕЙСИКе, работающей с многомерными массивами, вставляется обязательный участок текста. Он предназначен для организации подмены адреса подпрограммы, вычисляющей адрес элемента массива по его индексам, во всей откомпилированной программе на адрес введенной подпрограммы. Адреса начала и конца кода программы содержатся в ячейках с восьмеричными адресами 2002 и 2004 соответственно.

Метод подмены очень интересен, но возникает опасность, что в откомпилированной с БЕЙСИКа программе код 160056 (десятичный — 8146) является не только адресом подпрограммы, вычисляющей адрес элемента массива. Рассмотрим, какие могут быть варианты.

Во-первых, код 8146 может быть десятичной целочисленной константой 8146%. Этот вариант вполне вероятен, но область данных БЕЙСИК-программы обычно располагается вне диапазона адресов, содержащихся в ячейках 2002, 2004. Вследствие этого случай практически не опасен.

Во-вторых, код 8146 может быть частью текстовой строки с символами 'Ю.' (CHR$(&0340); CHR$(&056)). Такое тоже возможно, но практически безопасно по той же причине.

В-третьих, указанный код может быть машинной командой вычитания: SUB R0,@-(R6). Этот вариант возможен, но достаточно маловероятен из-за редкого способа адресации — косвенный режим автоуменьшения регистра программного стека. Скорее всего такая команда вообще не может быть сгенерирована компилятором БЕЙСИКа, но может быть включена внутрь программного кода программистом. Если же это случится, то программа работать не будет.

Избавиться от указанных опасностей достаточно сложно, так как адрес подпрограммы 160056 встречается в откомпилированном коде в различных вариантах окружения и различить, когда подмена адреса верна, трудно.

В целом программа В.В.Ермакова поучительна и полезна.

А.П.Гармашов,
программист

О текстах подпрограмм в кодах для БЕЙСИКа БК 0010

Загрузку программ в машинных кодах при работе с БЕЙСИКом на ПЭВМ БК-0010 обычно выполняют либо командой Бейсик-системы BLOAD, либо помещают тексты машинных кодов в операторы DATA, а оттуда — по нужным адресам. Перевод текста программы в кодах в операторы DATA для Бейсик-программы выполняется вручную.

Приведенная ниже программа предназначена для автоматической генерации операторов DATA из любых программ в кодах. Во время выполнения ее исходный текст заменяется строками программы на Бейсике с операторами DATA, содержащими представления кодов в десятичной, восьмеричной или шестнадцатиричной системах счисления. Текст программы генерации теряется.

Для пояснения работы программы сначала рассмотрим особенности интерпретатора ПЭВМ БК-0010 — интерпретатора "компилирующего типа", хранящего во время выполнения программы исходный текст и исполняемый модуль. Исходный текст начинается с адреса &03052, причем строки программы хранятся без номеров и не всегда в порядке их следования в программе. Сначала расположена наиболее "старая" строка с учетом исправлений текста, а последней — последняя введенная или измененная строка. В начало строки добавляется символ с кодом 0, если только она не начинается с зарезервированного слова БЕЙСИКа. В этом случае код 0 не добавляется, а слово заменяется внутренним кодом БЕЙСИКа. Например, для оператора DATA код &0100 (символ '@'). В конец строки всегда добавляется признак конца строки — символ с кодом &012. Номера строк хранятся отдельно в виде упорядоченного по возрастанию номеров списка, начиная с адреса &036754 в направлении уменьшения адресов. Для каждой строки программы выделено три слова. В слове со старшим адресом записан номер строки, затем адрес начала "тела" строки и адрес внутри исполняемого модуля (если программа уже транслировалась), иначе в третье слово записан 0. Отметим, что для Бейсик-системы важна информация в слове по адресу &02024.

Ю.Д.Усенков

О некоторых периферийных устройствах для БК 0010

Как известно, хорошие компьютеры имеют не только клавиатуру, дисководы и дисплей, т.е. те части, без которых работа на компьютере просто невозможна. В набор обычных для них периферийных устройств входят сегодня также принтер, джойстики, "мыши", сканеры (устройство в виде телекамеры для считывания изображений) и многое другое.

Все перечисленные дополнительные устройства для работы компьютера, конечно, необязательны и служат для облегчения работы программиста (пользователя). Владельцы БК, как оказалось, современными периферийными устройствами тоже не обделены. Уже несколько лет в фирменном магазине-салоне "Электроника" имеются в продаже джойстики и "мыши". Я хочу предложить читателям журнала некоторые сведения об этих устройствах.

1. **Джойстик.** В дословном переводе с английского- "веселая палочка". Пульт управления ПУ-1 типа джойстика, как это устройство названо в руководстве по эксплуатации, предназначен для управления играми на персональных и бытовых компьютерах, а также для управления курсором на экране и работы с графическими пакетами. Один из вариантов имеющихся в продаже джойстиков представляет собой настольное устройство в пластмассовом корпусе. Джойстик оснащен так называемой анатомической ручкой с четырьмя дополнительными кнопками. Для обеспечения надежного крепления джойстика к столу в нижней части его корпуса установлены четыре присоски. При установке джойстика на стол нужно с некоторым усилием прижать его корпус к поверхности стола, при этом присоски

Там хранится текущее значение указателя на конец списка строк (адрес последнего занятого слова в списке). До ввода текста программы это &036756.

Теперь можно рассмотреть непосредственно текст программы.

```
5 CLEAR 240
10 ? CHR$ (140) + CHR$ (140);
20 DATA &012501, &011502, &013703, &0352,
&0112223, &077102,
      &010337, &0352, &0207
30 DATA &011505, &04737, &0100536, &0207
40 DEF FNC$ (X%) = MID$ (STR$
(X%),2%,LEN(STR$ (X%))-2%)
50 DEF USR1 = &0400
60 DEF USR2 = &0422
70 FOR I% = 0 TO 12
80 READ J%
90 POKE &0400 + 2%*I%,J%
100 NEXT
110 B$ = CHR$(159)
120 ? B$ + "ГЕНЕРАТОР USR П/П ДЛЯ БЕЙСИ-
КА"
130 ? "РЕЗУЛЬТАТЫ: " + B$ + "1-DEC, 2-OCT, 3-
HEX ?";
140 T$ = INKEY$
150 IF T$<"1" OR T$> "3" THEN 140
160 ? T$
170 ? "ЗАГРУЗКА П/П В КОДАХ"
180 A% = USR2(&042000)
190 CN% = 0
200 POKE &0352, &03052
210 TB% = &036754
220 JJ% = &042000 + PEEK(&0266)
230 J% = 0
240 IF J% MOD 10% < > 0% THEN 300
250 A$ = "" + "@"
260 CN% = CN% + 10%
270 POKE TB%, CN%
280 POKE TB%-2%,PEEK(&0352)
290 TB% = TB%-6%
300 AD% = &042000 + 2%*J%
310 R% = PEEK(AD%)
320 IF T$ = "1" TH B$ = "" + FNC$(R%)
330 IF R%<0% THEN B$ = "" + "-" + B$
340 IF T% = "2" THEN IF R%<8% AND
R% > = 0% THEN B$ = "" + FNC$(R%)
      ELSE$B  = "" + "&O" + OCT$(R%)
350 IF T% = "3" THEN IF R%<10% AND
R% > = 0% THEN B$ = "" + FNC$(R%)
      ELSE B$ = "" + "&H" + HEX$(R%)
360 A$ = "" + A$ + B$
370 IF (J% + 1%) MOD 10% < > 0% AND JJ%-
AD%»0% THEN 420
380 A$ = "" + A$ + CHR$(10%)
390 P$ = USR1(A$)
400 A$ = ""
410 IF JJ%-AD%»0% THEN 430 ELSE 450
420 A$ = "" + A$ + ","
430 J% = J% + 1%
440 GOTO 240
450 POKE &02024,TB% + 2%
460 CLS
```

Строка 10 — начальная установка экрана. Начало служебной строки по-

---

надежно его фиксируют.

Габаритные размеры устройства: 120x100x180 мм.

Масса не более 300 граммов.

Порядок работы с ним следующий: управление игровым процессом на экране и изменение направления перемещения курсора осуществляется путем наклона анатомической ручки управления вправо, влево, вниз или вверх соответственно. При этом наклон ручки соответствует простому нажатию на клавиши перемещения курсора:--, --, I, I, т.е. данный джойстик не является пропорциональным координатным устройством. Кроме того, на нем имеются четыре дополнительные кнопки: две на анатомической ручке (под указательным пальцем и сверху) и две на самом корпусе. Эти кнопки могут выполнять заранее запрограммированные действия, например выстрел в игровой программе.

К сожалению, данный джойстик сопрягается с ЭВМ посредством стандартного семиштырькового разъема СШ-7, тогда как порт БК 0010, к которому его нужно подключать, имеет разъем-"гребенку".

Поэтому для работы с джойстиком на БК 0010 необходимо произвести некоторую его доработку: сделать переходник или просто перепаять разъемы. Таблица распайки выводов устройства на семиштырьковом разъеме:

Операция	Контакт	Маркировка провода штеккера СШ-7
Вперед	5	Красный
Назад	4	Синий
Влево	1	Зеленый
Вправо	3	Красно-синий
Стрельба	7	Красно-зеленый
Общий	2	Без маркировки

Из этой таблицы виден и второй "минус" данного джойстика: все четыре его кнопки распаяны на один и тот же вывод разъема, т.е. при работе программы, ис-

сле этого всегда имеет адрес &040000. В строках 20 и 30 — 2 функции в кодах. Они определены в строках 60, 70. Функция USR1 копирует содержимое символьной переменной (параметра), помещая копию с адреса, записанного в &0352. После этого &0352 указывает на первый свободный байт за копией. Функция USR2 вызывает программу монитора чтения с магнитофона. Параметр подпрограммы типа INTEGER указывает на начальный адрес загрузки.

После запроса имени загружаемой программы в кодах в строке 180 указан начальный адрес &042000 (в качестве буфера используем экран). Переменная CN%-счетчик строк генерируемого текста. Он изменяется через 10. В строке 200 для функции USR1 задается начальный адрес, куда будут помещаться строки программы (без номеров). С адреса TB% = &036754 в сторону уменьшения адресов формируется список вида: номер строки, ее адрес, затем пропускаем одно слово, номер следующей строки и т.д. В строках 240-440 в цикле считываются коды из буфера (экрана), формируется строка с оператором DATA и текстом кодов через запятую. Размер исходной программы в кодах читается из &0266. При формировании строки вместо DATA записывается символ '@', а в конце строки добавляется символ с кодом &012. В зависимости от выбора основания системы счисления (10,8,16) в строках 310-360 получается текст считанного в строке 310 машинного слова. Определенная в 40 функция FNC$ удаляет пробелы в начале и конце строки символов, получаемых функцией STR$. Но при отрицательном аргументе символ '-' будет потерян. Строка 330 добавляет '-' для этого случая. После накопления десяти команд в DATA либо окончания обработки всего буфера в строках 370-410 готовые строки по USR1 записываются, причем &0352 всегда указывает на следующий свободный байт. В строках 290-300 эта информация заносится в список строк Бейсик-программы. По завершении генерации управление передается в строку 450 для записи адреса конца списка строк в &02024.

Потеря исходного текста программы на этапе выполнения может использоваться и с другими целями, например для занесения на место текста данных или программ в кодах. Возможен и промежуточный вариант между сохранением и потерей исходного текста — частичное его из-

---

пользующей джойстик, нажатие на любую из четырех кнопок дает один и тот же результат, т.е. три кнопки из четырех, в общем-то, бесполезны. Правда, срок гарантии на джойстик невелик (всего несколько месяцев), и можно после окончания срока гарантии аккуратно вскрыть корпус и перепаять контакты кнопок на разные выводы порта БК-0010 (при этом придется заменить имеющийся у джойстика соединительный кабель другим, с большим числом жил).

2. **"Мышь"** — устройство ввода координатное (УВК-1).

Предназначено для расширения потребительских свойств ПЭВМ БК. Устройство является средством ввода графической информации в ЭВМ, а также позволяет перемещать курсор или элемент изображения по экрану дисплея, вводить данные в ЭВМ и использовать графику делового назначения.

УВК-1 выпускается специально для ЭВМ БК-0010 и БК-0010.01 и подключается к разъему порта БК с помощью многоштырьковой вилки-"гребенки". Таким образом, в отличие от джойстика, никакой переделки устройства для его подключения к БК не требуется.

Примечание В некоторых компьютерах необходимо внести изменения в их внутреннюю схему соединений с портом — адрес организации, выполняющей эти изменения, дан в руководстве по эксплуатации.

"Мышь" представляет собой небольшое устройство с вкладным шариком. Его катают по гладкой поверхности стола (максимально допустимая скорость перемещения 0,2 м/с), при этом движения курсора повторяют движения корпуса "мыши" (разрешающая способность 0,5 мм). В руководстве по эксплуатации дана тестовая программа для графического ввода рисунка в БК с помощью "мыши". Из нее легко можно понять, каким образом производится передача данных из устройства в ЭВМ.

При рисовании с помощью "мыши" переход в

менение. Так для хранения программы в кодах непосредственно в тексте Бейсик-программы можно разместить коды внутри символьной переменной. Для этого первая строка программы должна быть вида

10 AA$ = "12345..."

В этой строке форматируется символьная переменная, которая будет использована для хранения подпрограммы. Длина переменной — не меньше длины программы в кодах. Этой записи с адреса &03052 будут соответствовать коды: 0, потом текст AA$ =", содержимое переменной, " и код 12. Содержимое переменной AA$ записано с адреса &03060 (имя состоит из двух символов и взято для того, чтобы этот адрес был четным, т.е. границей слова). С этого адреса можно загружать программу в кодах, например, с помощью USR2, описанной выше, или просто операторами POKE. В результате строка 10 будет содержать необходимую информацию, а функцию пользователя определяем с адреса &03060. Изображение текста строки на экране имеет своеобразный вид. Теперь эту строку можно использовать в Бейсик-программах. Существенными ограничениями для такого способа размещения программы являются:

- программа в кодах должна быть перемещаемой, иначе ее надо будет скопировать по нужным адресам;
- программа не должна содержать код &012, поскольку он будет принят Бейсик-системой за признак конца строки.
- длина программы ограничена размерами символьной переменной.

Таким образом, несмотря на компактность, этот способ может иметь ограниченное применение в основном для коротких программ. Любопытными могут быть побочные эффекты при отображении на экране текста программы с такой строкой. Например, при размещении кодов из строки 30 в символьной переменной соответствующая строка видна на экране в виде двух символов, а если разместить слово &0112414, то на экране по команде LIST вообще будет только графический курсор.

Программа — генератор текстов операторов DATA, описанная в статье, имеет ограничение: длина программы в кодах (исходные данные для генератора) не должна бать более &0212. Опыт работы показал, что при больших размерах начинаются сбои при формировании переменной A$.

Кроме того, привожу краткое

режим записи графической информации производится кратковременным нажатием на клавишу "Л", выход из режима записи-повторное нажатие "Л". Аналогично переход в режим стирания — клавиша "П", выход-повторное нажатие "П".

При одновременном нажатии клавиш "Л" и "П" происходит выход из графического режима.

Питание УВК: напряжение +5,0 ± 0,5 В; максимальный ток 180 мА; габаритные размеры 80x115x50 мм

Масса не более 0,25 кг.

Длина присоединительного кабеля 1000 мм.

Примечание. Вообще-то, "мышь" не обязательно катать по столу. Можно перевернуть ее вверх ногами и просто рукой поворачивать шарик. При этом для работы с "мышью" требуется меньшее пространство.

Цена устройства "мышь" 185 рублей, цена джойстика 51 рубль.

По рекламе все в том же магазине "Электроника" появились: дисковод (цена около 980 рублей) и принтеры (цена от 1500 до 3000 рублей).

И еще. Хочу поделиться с читателями журнала опытом нестандартного использования интерпретатора Бейсик. С его помощью можно не только составлять и просчитывать программы. Предположим, нужно какую-то информацию (например, куски текста) записать на магнитную ленту, причем каждый блок информации имеет свой порядковый номер и все эти блоки у вас написаны "вперемешку". Тогда можно вводить эти блоки текста вместо строк программы подряд, начиная с порядкового номера. Компьютер расставит блоки текста в порядке возрастания номеров и позволит записать текст на магнитофон, пользуясь теми же командами, которые используются для записи программ. То, что введенные строки текста не являются строками программы, БК просто "не почувствует". (Кроме, конечно, попыток запустить такую лжепрограмму на выполнение.)

описание еще одной программы, на этот раз в кодах.

Печать строк при выполнении программы на БЕЙСИКе БК 0010.

Команда LIST БК 0010 не реализована при работе в программном режиме. Предлагается программа в кодах для вывода на экран строк Бейсик-программы во время выполнения. Программа оформляется как функция пользователя USR и может загружаться с любого адреса.

Параметр функции — номер вводимой строки (целое число).

```
13701   2026    24115   1405    20137   2024    3420    24141
771     14103   105723  1407    12702   133400  153102  11201
12702   20337   104020  10301   12702   5377    104020  207
```

Нам пишут

Жариков Л.Н.

Программа для программатора ППЗУ и ее оптимизация

Многие читатели заинтересовались опубликованной в выпуске № 2 за этот год программой Р.Барташюса для программатора ППЗУ и предлагают свои изменения. Вызвало нарекания большое количество опечаток-неизменных спутников любой сложной публикации.

Вашему вниманию предлагается обзор предложений. Свой вклад в программу внесли: В.Г.Бендаринов из г.Горького, Г.Мухамедов из г.Ташкента, Д.Ю.Усенков из Москвы, знакомый вам по публикации в том же выпуске, и автор данной статьи.

Сначала ответы на некоторые вопросы. Многих читателей удивило, что в строках 5050,5060 исходного текста вычисления не выполнены самим автором программы. Очевидно, эти строки специально публиковались Р.Барташюсом в исходном виде для того, чтобы подчеркнуть аддитивные составляющие передаваемых в порт данных, а сложение двух чисел он доверил выполнить читателям.

Вызвало недоумение пользователей и наличие пустых операторов PRINT. Ведь разместить выводимую на экран информацию в Бейсике легко с помощью инструкций AT, указывающих позицию. Думается, что это было сделано автором для большей простоты и наглядности текста.

Теперь об исправлениях программы, которые предлагаются с целью улучшения ее пользовательских качеств и стиля программирования. Нужно сказать, что указанные читателями изменения и дополнения касаются только отдельных сторон ее оптимизации и не дают максимального эффекта, который хотелось бы получить.

Если же действительно говорить о существенной оптимизации программ, то это достигается применением сочетания достаточно стандартных способов.

Вот некоторые из способов оптимизации:

- тщательный выбор метода и самого алгоритма (обычно это дает максимальный эффект);
- выбор типов и структуры данных, которые обеспечивают оптимизацию соответствующих параметров программы (время исполнения, объем занимаемой памяти);
- вычисление везде, где только возможно, числовых выражений;
- раскрытие и другая оптимизация циклов;

- выделение подпрограмм или, наоборот, их явная подстановка в места, критичные по времени выполнения;

- выбор форм операторов языка, обеспечивающих наиболее лаконичное выполнение функций или наименьшее время выполнения;

- использование специфических особенностей реализации языка конкретной ЭВМ;

- написание наиболее критичных по времени участков программы в машинных кодах;

- жертвование сервисом ради уменьшения объема программы.

Способы приведены приблизительно в той последовательности, в которой их следует применять. Как знают опытные программисты, на практике применение этих способов обычно начинается в обратном порядке, приводя к значительным затратам усилий для достижения того же эффекта, который дает правильная последовательность.

На примере вызвавшей специфический интерес к программе для программатора ППЗУ, используя некоторые предложения читателей, покажем, как применять приведенные выше приемы. Будем пользоваться текстом программы Р.Барташюса, приведенным во втором выпуске ВТ.

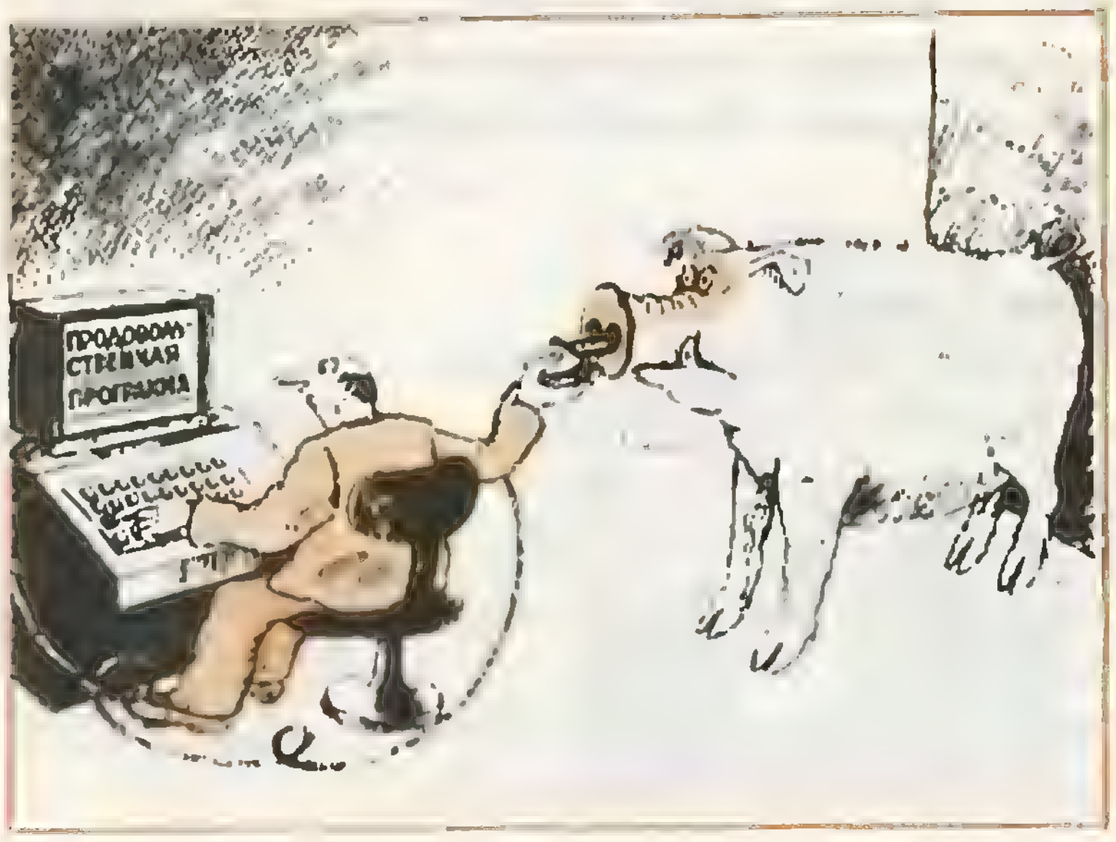
Первый в перечне способ ни у кого не вызывает сомнений, если это связано с достаточно солидной программой. Имея перед глазами работоспособную версию, метод и саму структуру алгоритма, будем считать пока правильными, как это делают сначала все программисты, и используем второй метод.

Во второй строке исходной программы описан массив D(63,15), по умолчанию являющийся массивом двойной точности. Каждый элемент такого массива занимает в памяти 8 байтов. Если мы вспомним, что данный массив служит для хранения прошиваемых в ППЗУ или загружаемых из ППЗУ байтов, то восьмикратный излишек занимаемой этими данными памяти вызовет некоторое удивление. Для хранения байта, обычно достаточно одного байта памяти!

К сожалению, в Бейсике нет типа данных 'byte', как, например, в языке Турбо-Паскаль. Но нас вполне может устроить описание целочисленного массива D%(63,15), говорящее о том, что каждому элементу будет выделено два байта памяти. С этим можно пока смириться. Все-таки высвобождается 6 Кбайт памяти! Правда, этот выигрыш несколько уменьшится за счет хранения в исходном тексте знаков '%'.

Достичь еще большего выигрыша памяти можно, введя вместо массива D%(63,15) массив символьных строк D$(15) и упаковывая в процессе расчетов байты данных в соответствующие символы одной из шестнадцати строк. Но мы этим пока заниматься не будем. Анализ также показывает, что вообще необходимости в числовых переменных нецелого типа в этой программе нет. Поэтому все переменные заменим на переменные целого типа, добавив к ним символ '%'. Этим действием будет сэкономлена память и увеличено быстродействие, так как операции с целыми числами выполняются существенно быстрее.

Точно таким же образом можно заменить и все числовые константы на константы целого типа, проявляя, однако, известную осторожность, так как диапазон целых чисел в Бейсике достаточно узок [-32768, 32767]. Вследствие этого написать в строках 4040, 5050 и 5060 выражение типа



```
...,N%-32768%...
```

вам не удастся. Нужно будет вводить, например, так:

```
...,N%-32767%-1%... ,
```

или так

```
...,N% + &0100000... ,
```

или так

```
...,N% + &H8000... .
```

Ко второму методу оптимизации относится и удаление лишних переменных. Посмотрев на программу несколько внимательнее, можно заметить, что переменная 'B%' в строках 4110, 4120, 4160, 7000, 7050, 7060 используется только для передачи данных подпрограмме в строке 2000. Если от этой весьма короткой подпрограммы избавиться (тоже способ!) и попутно заменить операторы PRINT на знаки вопроса, то соответствующие места будут выглядеть короче:

```
...
4110-4130 - удалить
4140 ? "ошибка по адресу" ; HEX$(I% + KF%)
...
4150-4170 - удалить
4180 ? HEX$(J%) ; "H"
...
7000-7010 - удалить
7020 ? AT(0%,5%) ; HEX$(I%)
...
7050-7070 - удалить
7080 ? AT(J%*3% + 6%,5%);HEX$(D%(I%-
KF%) AND &HFF)
...
```

Вычислим и преобразуем теперь, где это возможно, числовые выражения. Таких мест два: строки 4030, 4040 и 5040-5080. Так как эти участки находятся внутри циклов, то это связано и с оптимизацией циклов. Первый цикл после вынесения из него вычислений, не связанных с переменными цикла I% и J%, будет выглядеть так:

```
...
4000 ? "Ждите..."
4005 K% = KF%*16% + &H8000
4010 FOR I% = 0% TO 63%
4015 L% = I%*16% + K%
4020 FOR J% = 0% TO 15%
4030 N% = L% + J%
4040 POKE - 52%,N%
4050 IF CO% = 1% GOTO 4100
4060 D%(I%,J%) = PEEK(-52%)
4070 NEXT J%,I%
4080 - удалить
...
```

Второй цикл изменится аналогично:

```
...
5000 ? "ЖДИТЕ..."
5010 K% = KF%*16% + 2048%
5020 FOR I% = 0% TO 63%
5025 L% = I%*16% + K%
5030 FOR J% = 0% TO 15%
5040 N% = L% + J%
5045 FOR W% = 0% TO 5%
5050 POKE -52%, -30720%
5060 POKE -52%, N% + &H8000
5070 POKE -52%, N%
5080 POKE -52%, D%(I%,J%) + 30720%
5090 FOR T = 0 TO 50
5100 NEXT T,W%,J%,I%
5110-5130 - удалить
...
```

Заметьте, цикл по W% внесен внутрь циклов по I% и J%. Этим достигается уменьшение количества вычислений вспомогательных переменных K% и L%.

В результате циклы будут выполняться быстрее. Чтобы дополнительно сэкономить еще, в операторах POKE и PEEK можно замениь целочисленные константы на шестнадцатеричные или восьмеричные. Для перевода в другую систему счисления можно воспользоваться функциями OCT$ и HEX$, введенными в непосредственном режиме, например,

```
? OCT$(-52)
177714
Ok
```

или

```
? HEX$(-52)
FFCC
Ok.
```

Запустив исправленную таким образом программу на исполнение командой RUN, убедимся, что в результате оптимизации цикл чтения из ППЗУ сократился до 3-4 с, а цикл записи почти не изменился. Попутно отметим, что для проверки совсем не обязательно иметь аппаратную часть программатора ППЗУ, достаточно вставить блок нагрузок в разъем параллельного порта при выключенном питании БК. Это обеспечит чтение

упорядоченных данных в массив D% в режиме ввода из ППЗУ и режиму контроля будет что проверять. Записать данные в блок нагрузок, естественно, не удастся.

Дальнейшая надежда уменьшения времени выполнения записи в ППЗУ связана с самим внутренним циклом в строке 5090. К сожалению, сокращение времени исполнения этого пустого цикла, введенного для организации определенной задержки, связано с риском увеличения вероятности возникновения ошибок прошивки ППЗУ. Не зная технических параметров схемы программатора ППЗУ, уменьшать это время не стоит.

К сведению любопытных читателей сообщим, что цикл 5010-5130 без операторов 5050-5080 выполняется 8 мин 05 с, а измененный цикл 5010-5100 вместе с операторами РОКЕ — 8 мин 21 с. Таким образом, дальнейшая оптимизация рассматриваемого участка практически нецелесообразна.

Чтобы еще более сократить количество строк, заменим вызовы подпрограммы в строках 3340, 3360, 6110, 7030 на сам оператор ?CHR$(155%), переключающий дисплей в режим 64 или 32 символов в строке. В результате строку 7030 можно удалить, а строка 7020 примет вид

```
...
7020 ? AT(0%,5%);HEX$(I%);CHR$(155%)
...
```

Окинем взглядом всю программу целиком (см. в конце статьи). Можно заметить, что выделяются несколько в чем-то однородных участков программы: 70-220, 240-290, 1080-1130, 3080-3130.

Попробуем несколько сократить эти участки, используя специфические особенности реализации Бейсика на БК0010.01 и выбирая формы операторов, обеспечивающие наиболее лаконичное исполнение программой возложенных на нее функций. При этом надо помнить, что включение в текст уникальных возможностей версии языка, равно как и подпрограмм в машинных кодах, ограничивает диапазон компьютеров, на которых программа будет работать без переделки.

Первый из рассматриваемых участков написан нерационально, но очень нагляден. Можно его сжать, если указать строки и колонки, с которых начинать вывод меню, инструкциями AT и LOCATE.

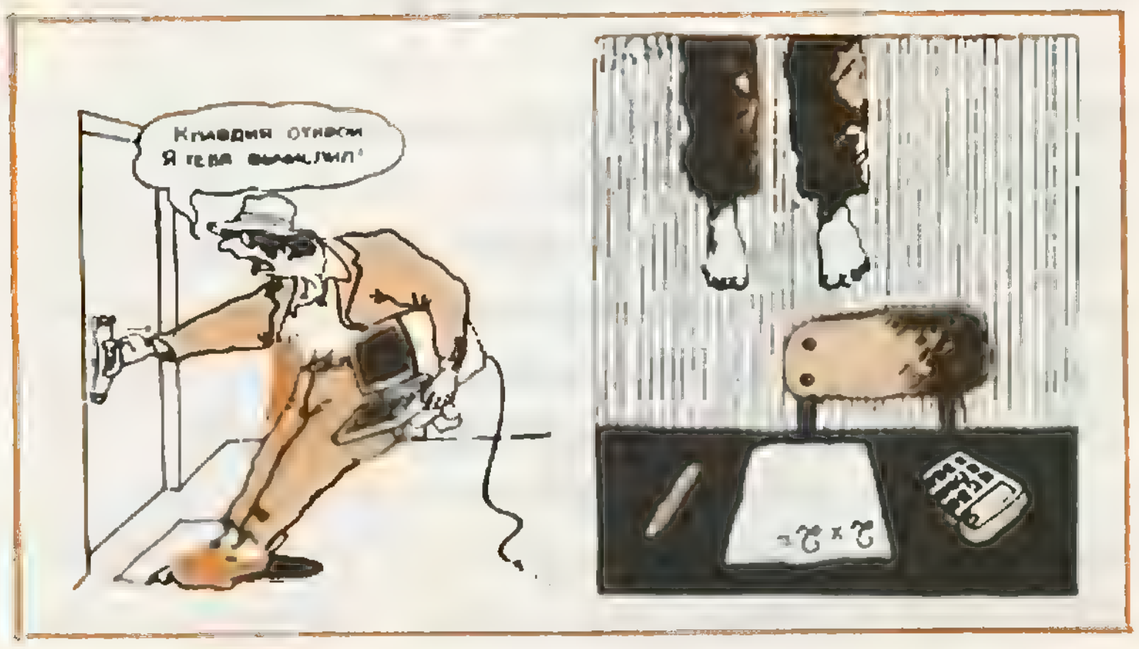
Введение операторов выбора ON ... GOSUB [GOTO] ... позволяет сократить строки 240-290 второго участка. В результате текст будет выглядеть следующим образом:

```
...
70 IF KF% = 0% THEN ? "К573РФ1 (IРФ2)" ELSE
? "К573РФ2(II)"
80, 90, 110, 130, 150, 170, 190, 210, 220 — удалить
100 ?AT(0%,3%);"1 ввод с клавиатуры"
120 ?AT(0%,5%);"2 ввод из ППЗУ"
140 ?AT(0%,7%);"3 вывод в ППЗУ"
160 ?AT(0%,9%);"4 корректировка"
180 ?AT(0%,11%);"5 РФ1(IРФ2) или IIРФ2"
200 ?AT(0%,13%);"6 контроль"
230 INPUT T$
240 ON VAL(T$) GOSUB 6000,4000,5000,3000
250 ON VAL(T$)-4% GOTO 310,330
260-290 - удалить
...
```

Третий участок 1080-1130, предназначенный для распознавания цифр шестнадцатиричного двухзначного числа, можно сжать, получая значения кодов вводимых символов с помощью функции ASC:

```
...
1080 IF ASC(Y$)»70 OR U%««0% OR Y$ = "0"
GOTO 1140
1090 C% = ASC(Y$)-55%
1010-1130,1160 - удалить
...
```

Четвертый участок программы 3060-3130 после анализа может быть



сокращен за счет введения распознавания некоторых подмножеств значений переменной R% и объединения условных операторов. Например,

```
...
3080 IF FIX(SQR(R%)) = 5% THEN ON R%-24%
GOTO 3210,3170,3190
3090 IF R% = 81% THEN 3360 ELSE IF R% = 67%
THEN 3140 ELSE IF R% = 8% THEN 3270 ELSE
3060
3100-3130 - удалить
...
```

Пытливые читатели могут поупражняться в поиске самого лаконичного способа ветвления алгоритма для данных значений R% (25,26,27,81,67,8). Заметьте, что в результате вместо шести программных строк осталось две, но структурность программы не нарушилась.

Теперь можно заняться и сервисом. Правда, при наличии свободной памяти жертвовать мы им пока не собираемся.

Читатель В.Г.Бендаринов (г.Горький) предложил набросок начального участка программы для несколько другой организации меню. Перемещение по меню вверх-вниз осуществляется с помощью клавиш управления курсором, а выбор темы — нажатием клавиши <ввод>. Работа с таким меню более комфортна.

С некоторыми исправлениями в предлагаемом наброске начальный участок программы будет выглядеть следующим образом:

```
20 DIM D%(63%,15%)
30 SI% = 0%
35 Y% = 43% "Вертикальная координата строки меню"
40 CLS
50 CO% = 0%
60 KF% = SI%*64%
70 IF KF% = 0% THEN ? "K573РФ1 (IРФ2)" ELSE
? "K573РФ2(II)"
90 LOCATE 5%,5%,0% ' убрать курсор
100 ?"Ввод с клавиатуры"
120 ?AT(5%,7%); "Ввод из ППЗУ"
140 ?AT(5%,9%); "Вывод в ППЗУ"
160 ?AT(5%,11%); "Корректировка"
180 ?AT(5%,13%); "РФ1(IРФ2) или IIРФ2"
200 ?AT(5%,15%); "Контроль"
210 ?AT(5%,17%); "Выход"
220 Y1% = Y%
221 LINE (30%,Y%) - (200%,Y% + 20%),1%,B
'Нарисовать рамку темы меню
222  C$ = INKEY$
223 IF C$ = "" THEN 222 ELSE T% = ASC(C$)
'Между кавычками нет пробела
224 IF T% = 26% THEN IF Y%»43% THEN
Y% = Y%-20% ELSE Y% = 163%
225 IF T% = 27% THEN IF Y%«163% THEN
Y% = Y% + 20% ELSE Y% = 43%
226 IF T% = 10% THEN IF Y% = 163% THEN
STOP ELSE 230
227 IF Y% < > Y1% THEN LINE (30%,Y1%)-
(200%,Y1% + 20%),0%,B
Стереть рамку темы меню
228 GOTO 220
230 LOCATE 1%,1%,1% ' Восстановить курсор
240 ON (Y%-23%)/20% GOSUB
6000,4000,5000,3000
250 IF Y% = 123% THEN 310 ELXE IF Y%«»143%
GOTO 40
260 CO% = 1%
270 GOSUB 4000
280 ?„"Для возврата в меню - «ввод»" 'Обратите внимание на запятые
290 IF INKEY$ = " " THEN 290 ELSE 40 'Между
кавычками нет пробела
310 IF KF% = 0% THEN SI% = 1% ELSE
SI% = 0%
320 GOTO 40
1000 T% = 1%
...
```

Такая организация работы с меню будет полезна во многих программах читателей и обеспечит большую наглядность выбора.

Можно заметить, что в режимах ввода с клавиатуры и редактирования над вводимой строкой отсутствует нумерация вводимых байтов, облегчающая установку нужной позиции. Такую нумерацию сделать просто. Для этого нужно ввести строку

```
...
7075 ? AT (J%*3% + 6%,3%); HEX$(J%)
... ,
```

а в строках 3320, 7020, 7080, 7110, 8000, 8020 номер строки экрана заменить на 5%. Работать станет удобнее.

Заметим, что наша оптимизация рассматриваемой программы не была всеобъемлющей: мы не применяли самый эффективный, но и самый сложный способ оптимизации, касающейся уточнения метода и алгорит-

ма. Часто этот прием приводит к проектированию программ заново, поэтому имея уже работоспособную программу, все программисты прибегают к нему в крайнем случае. В начале нашего совместного пути по программе мы приняли авторскую структуру программы и предложенное им разделение на режимы работы. Теперь выделим читателям для проработки несколько идей.

Во-первых, имеется возможность объединения режимов ввода с клавиатуры и корректировки таблицы прошивки в один экранный режим редактирования. Диапазон адресов в микросхеме ППЗУ можно задавать явно.

Во-вторых, контроль правильности хотелось бы выполнять не только в отдельном режиме, но и непосредственно в цикле прошивки. В этом случае циклическое повторение прошивки необходимо только при обнаружении ошибки. Такое решение позволит сэкономить много времени и обеспечить адаптацию программы к особенностям конкретного корпуса микросхем ППЗУ. Дополнительно при неверной прошивке можно программно увеличивать длительность задержки.

Указанные два момента могут существенно повлиять на работу всей программы, как это обычно и случается при использовании первого способа оптимизации.

Одним из последних штрихов является выбор правильного шага перенумерации. На этом тоже можно сэкономить память БК. В начале работы опытные пользователи выбирают шаг перенумерации строк обычно равным 10, чтобы легко вставлять новые строки. Полуотлаженный вариант программы, особенно при недостатке памяти, можно нумеровать и с шагом 5. В публикациях обычно используют шаг 10.

В заключение нашей работы вставим в начало комментарии с наименованием программы автором и участвовавшими в ее совершенствовании читателями. Нумерацию строк программы оставим авторскую, так как при желании команда Бейсик-системы

```
RENUM 10,1,10
```

быстро выполнит нумерацию с шагом 10, а авторская нумерация позволяет легче выполнять исправления в исходной программе.

Окончательную версию программы записываем на магнитофон командой

```
CSAVE "PPZU" .
```

Подведем итоги нашей короткой экскурсии в мир оптимизации программ. Благодаря настойчивому труду исходный текст программы для программатора ППЗУ стал существенно короче, разнообразие палитры операторов языка шире, более 5 Кбайт памяти осталось для дальнейшего совершенствования программы, сама работа программы значительно быстрее, а диалог — удобнее и понятнее. К сожалению, таких результатов достичь удается далеко не всегда.

Терпеливые читатели, добравшиеся до конца и оптимизировавшие программу вместе с нами, получили достаточное количество информации, чтобы самим оптимизировать собственные разработки. При этом они споткнутся о множество подводных камней, которые мы старательно обходили при первом достаточно поверхностном знакомстве с оптимизацией программ. Хочется надеятся, что их труд не пропадет даром и программы, публикуемые в выпусках серии, будут гораздо рациональнее.

Прощаясь с внимательным читателем, обладающим чувством юмора, приведем несколько советов.

Полезные советы

1. Не берите лишнего в памяти! И вообще ничего лишнего не берите! Как мы видели, основной выигрыш по объему памяти и времени выполнения программы дал в нашем примере правильный выбор типа данных. Ключевой массив D%(63,15) стал целого типа и сэкономил сразу 6 Кбайт памяти. Когда экономия памяти не является главным элементом оп-

тимизации, можно пользоваться принципом умолчания.

2. Делайте ваш выбор! Выбирайте тщательнее, но быстрее! Языковая структура типа "выбор" (в Бейсике оператор ON) может сэкономить много операторов IF. Вложенные операторы IF тоже экономят память.

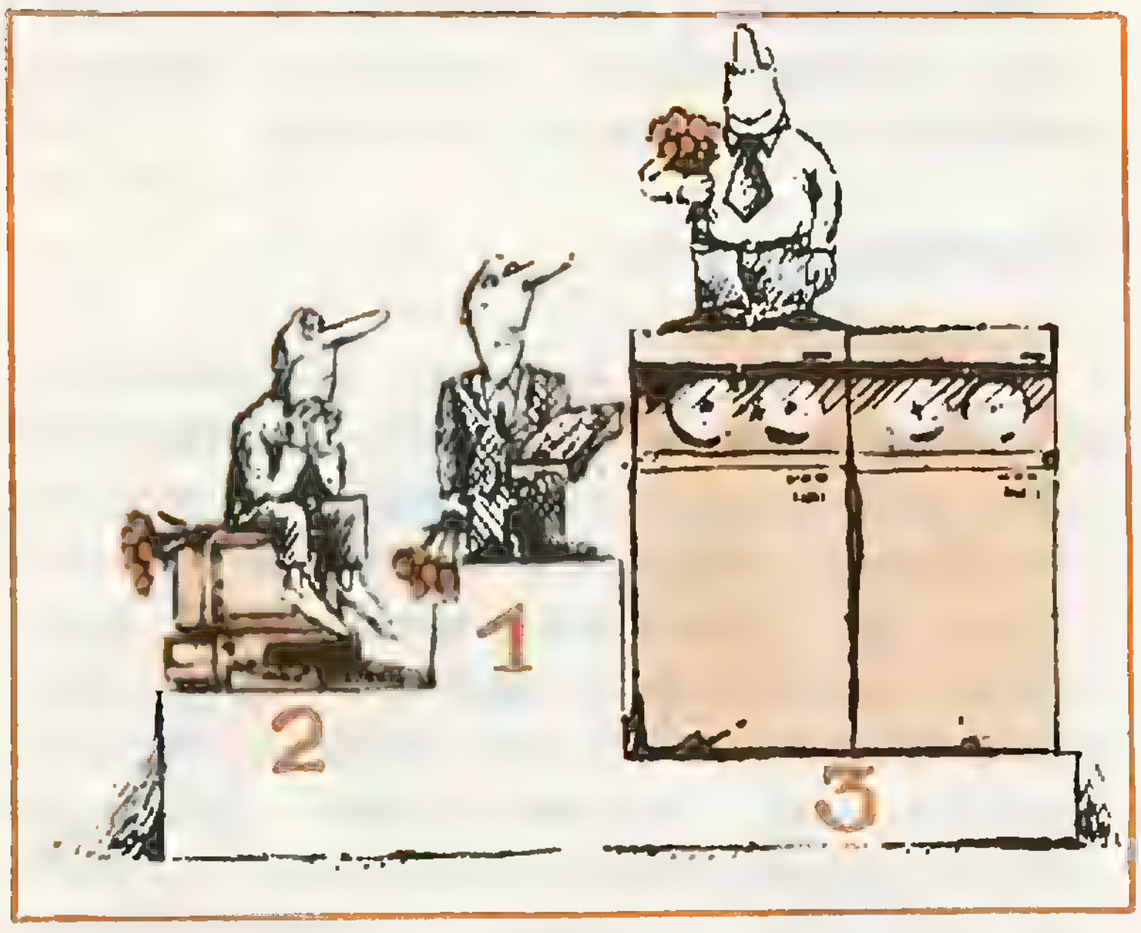
3. Чем меньше переменных, тем лучше! Введение лишних переменных не только расходует оперативную память, затуманивает алгоритм, но обычно уменьшает и скорость работы программы.

4. Выносите все, что можно! Не забудьте только внести обратно главное! Оптимизируя по времени выполнения программу, оптимизируйте тела циклов, вынося из них независимые от переменной цикла выражения.

5. Не полагайтесь на компьютер - вычисляйте сами все, что можно вычислить! Подставляйте результаты в программу. Для ясности программы в публикациях этого можно не придерживаться. Закручивайте формулы как можно круче!

6. Пользуйтесь при необходимости не только десятичными константами! Шестнадцатиричные, восьмеричные и двоичные константы часто экономят память и ускоряют процесс вычисления, делая программу еще более совершенной.

7. Оптимизируйте только отлаженные, часто публикуемые и используемые программы, другие сразу после использования бросайте в мусорную корзину! Оптимизация ненужных и неотлаженных программ нужна только для тренировки ваших навыков.



8. Подбирайте шаг нумерации строк программы. Помните, "шире шаг" — не для Бейсика БК0010.01.

9. Внимайте тщательно! Оптимизация требует внимания и времени. Тестируйте программу при любых самых незначительных изменениях! Самые очевидные вещи в программировании оказываются при ближайшем рассмотрении очевидно неверными. При первом же случайно выполнившемся прогоне программы считайте ее отлаженной и сдавайте заказчику.

10. Чем больше нетривиальных подсказок в диалоге, тем меньше вопросов у пользователя и оптимальнее интерфейс программы.

11. Создавая программу для БК0010, примите все меры к тому, чтобы она не прошла на IBM PC/XT/AT. Для этого вставляйте в программу только те операторы языка, которых нет в других диалектах. В крайнем случае врежте в программу кусок в машинных кодах. Это положит любого пользователя на лопатки, наверняка обеспечит непереносимость программы на другие ЭВМ и минимальное ее использование.

12. Публикуйте оптимальные программы, не проверяя на компьютере. Это избавит вас от сомнений. Читатели заодно и проверят!

```
1 REM Программа для программатора ППЗУ
2 REM *** БК0010.01 ***
3 REM Автор:
4 REM Р.Барташюс
5 REM Модификация:
6 REM В.Г.Бендаринов
7 REM Л.Н.Жариков
8 REM Г.Мухамедов
9 REM Д.Ю.Усенков
10 REM 1990г.
20 DIM D%(63%,15%)
30 SI% = 0%
35 Y% = 43% 'Вертикальная координата строки меню
40 CLS
50 CO% = 0%
60 KF% = SI%*64%
```

(Окончание см. на с. 33)

БК ЗА РОГА

В.В.Сапогов
(г.Чебоксары)

Система автоматического управления накопителем на магнитной ленте (бытовым магнитофоном) на базе БК-0010

Назначение и основные свойства

Система автоматического управления накопителем на магнитной ленте (бытовым магнитофоном) предназначена для работы в составе прикладных программ, реализованных на персональном компьютере БК-0010.01, написанных на языке БЕЙСИК.

Как известно, объем памяти, предоставляемый пользователю БК в режиме полноформатного экрана, составляет 16 Кбайт. За вычетом ОЗУ, используемого для служебных нужд Бейсик-системы, фактический объем ОЗУ, которым может располагать пользователь, составляет чуть более 14 Кбайт. Такой ограниченный объем ОЗУ приводит к тому, что массивы данных для пользовательских программ чаще всего не превышают 3—4 Кбайта, что во многих случаях неприемлемо мало. Одним из способов решения проблемы нехватки памяти для данных может быть последовательная обработка блоков информации, при которой исходные данные считываются с магнитной ленты и на нее же "сбрасываются" результаты вычислений или промежуточные данные, при этом единовременно обрабатываемый блок информации по величине ограничен объемом ОЗУ, а общий объем — емкостью кассеты с магнитной лентой и способом записи. Такой способ обработки позволяет существенно увеличить возможности БК.

Предлагаемая система автоматического управления накопителем на магнитной ленте позволяет выполнять программно-управляемые запись и считывание с магнитной ленты, а также автопоиск необходимых записей. Система включает в себя программную и аппаратную части.

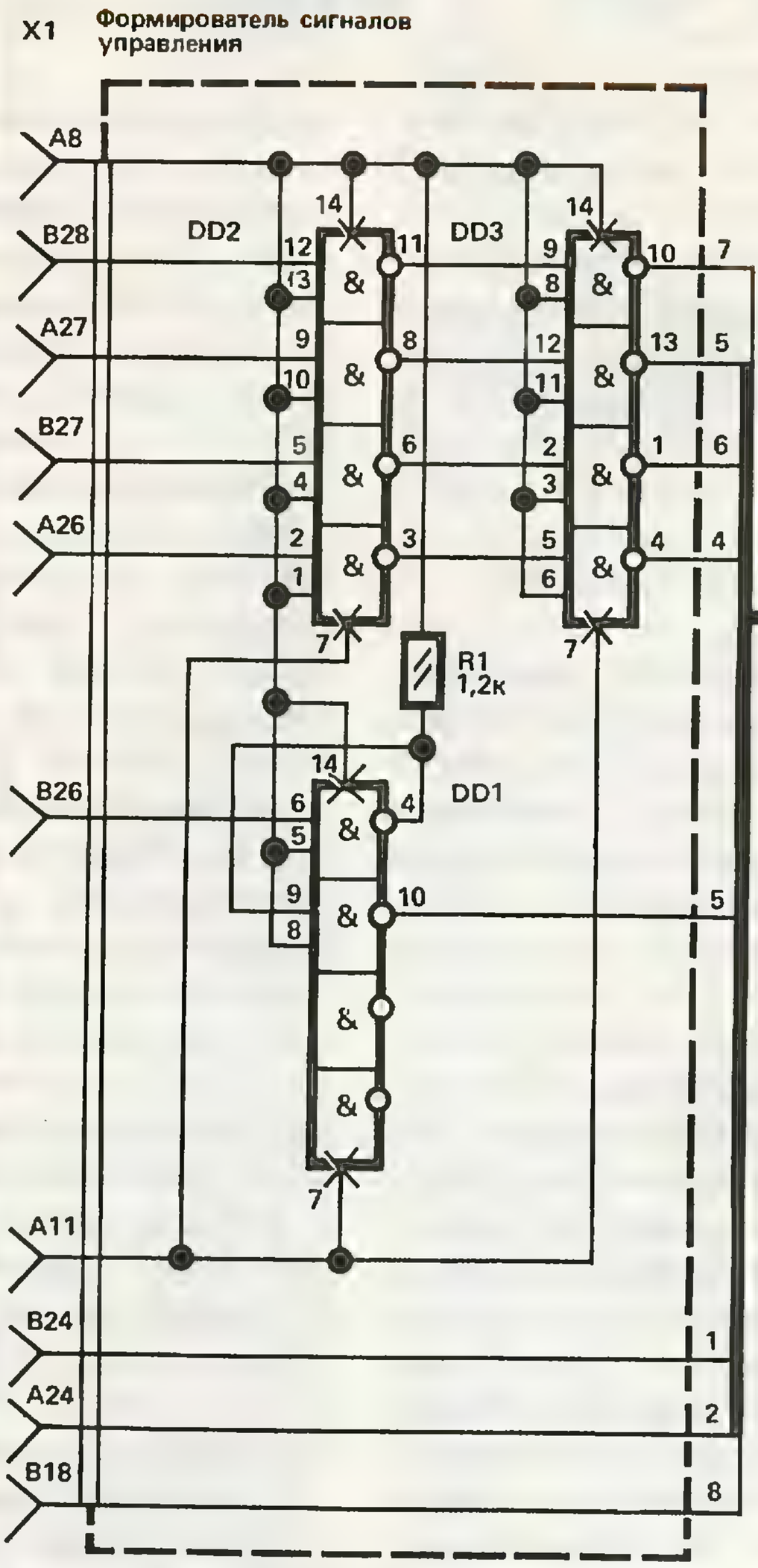
Программная часть системы фактически состоит из трех подпрограмм, первая из которых реализует программное выполнение операторов SAVE, BLOAD. (Эта программа целиком взята из статьи "Включение подпрограмм в машинных кодах в программы на БЕЙСИКе для БК0010", серия "Вычислительная техника и ее применение". — 1988. — № 9.) Вторая выполняет автопоиск нужного массива на магнитной ленте по заранее заданному адресу, третья выполняет вспомогательные функции только при подготовке программы к работе и не участвует непосредственно в работе остальных частей. Операторы всех программ должны быть вставлены в прикладную программу. Управление программами выполняется с помощью переменных, значения которых задаются перед вызовом подпрограммы.

В качестве накопителя на магнитной ленте необходимо использовать кассетный магнитофон, имеющий электронно-логическое управление и оптоэлектронный датчик, связанный с движением ленты. Автором система была смонтирована на магнитофоне "Нота-225".

Аппаратная часть системы позволяет реализовать режим работы магнитофона под управлением компьютера. Для управления магнитофоном используются шесть разрядов выходного порта и два разряда входного порта компьютера. Разряды выбраны таким образом, чтобы сохранить возможность подключения к компьютеру принтера, связь с которым осуще-

Компьютер
БК–0010.01

Разряд порта	Управляющий код	Назначение
		+5В
Вывод 9	512	перем. вправо
Вывод 10	1024	перем. влево
Вывод 11	2048	стоп
Вывод 12	4096	воспр.
Вывод 13	12288	запись
Ввод 0	1	опто-датчик
Ввод 1	2	авто-стоп



ствляется через те же порты. Связь компьютера с магнитофоном осуществляется посредством 8-жильного кабеля. В схеме аппаратной части используются доступные микросхемы серий К155 и К140.

Перед запуском системы в эксплуатацию после изготовления и монтажа плат аппаратной части необходимо выполнить определенные действия по настройке программы под конкретную программу пользователя.

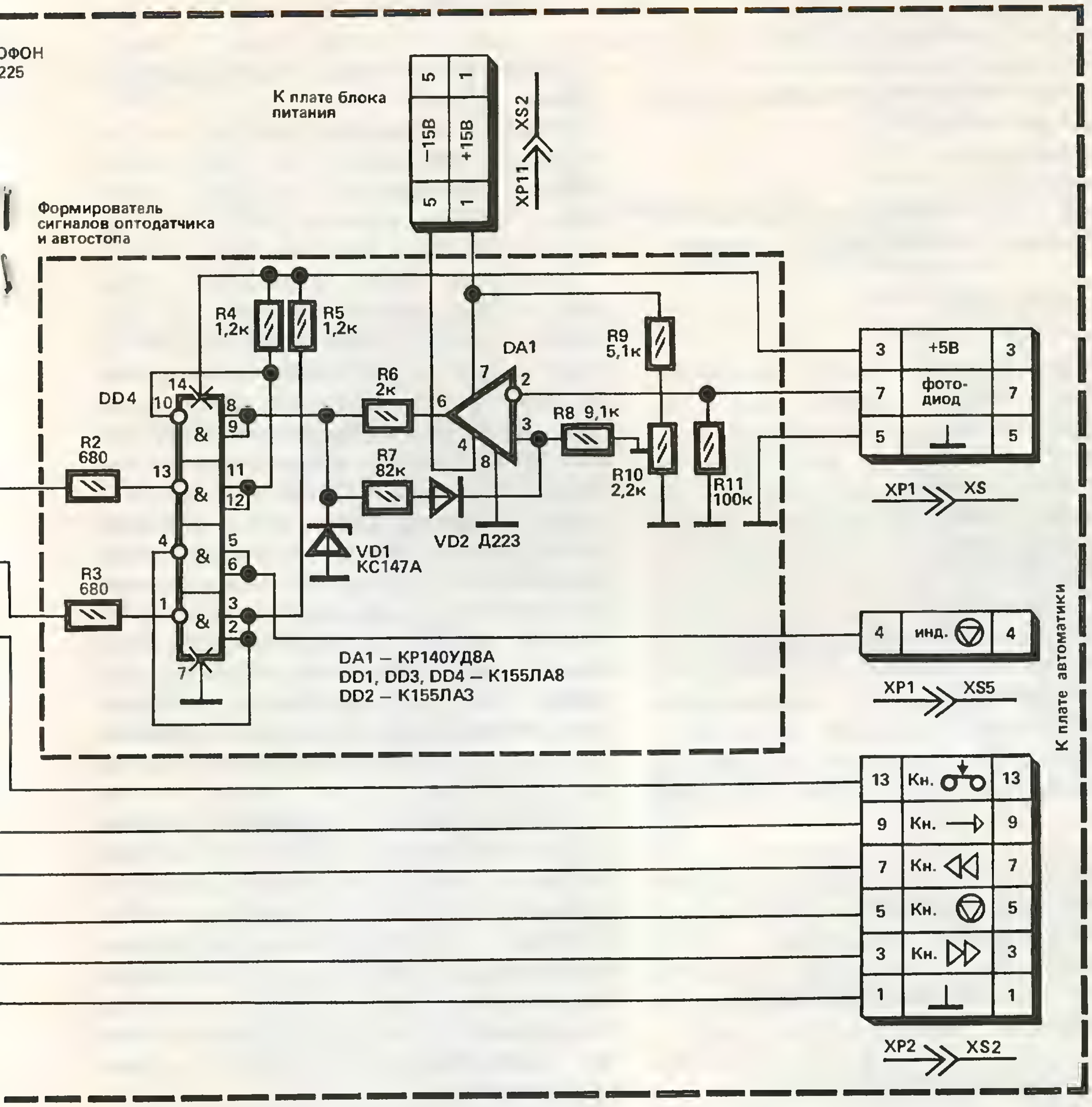
Автор полагает, что затраты труда и времени компенсируются удобством работы с программой после ее окончательной настройки и увеличением возможностей компьютера БК-0010.

Программное обеспечение

Текст программы, используемой в системе автоматического управления накопителем на магнитной ленте (бытовым магнитофоном).

```
210 DATA 5443, 4930, 5569, 214,-27503,
32450, 135
220 FOR I 1% = 0%TO6%
230 READ I%
240 POKE &0400 + 2%.I 1%, I%
```

ОФОН
225




```
250 NEXT I 1%
260 DEF USR1 = &0400
270 DEF USR2 = &0100602

5000 J% = &0177714
5002 I2% = 0%
5005 Z1% = 0%
5010 POKEJ%, 1024%
5030 GOSUB5990
5050 A1% = 0%
5060 IF INP (J%,&B10) = 0% GOTO 5060
ELSE POKEJ%,512%
5100 I1% = 1%
5110 IF INP(J%,&B1) = 0% GOTO 5100
5120 IF I1% < 1% GOTO 5110
5130 A1% = A1% + 1%
5140 IF A1% > = R1% GOTO 5160 ELSE
I1% = 0%
5150 GOTO 5110
5160 IF Z% = 1% THEN POKEJ%,12288%
ELSE POKEJ%,4096%
5170 A$ = USR1(NF$)
5180 IF Z1% = 1% THEN POKE&0322,J3%
ELSE POKE&0322,J1%
5190 IF Z% = 1% GOTO 5360
5200 POKE&0320,3%
5210 I1% = USR2(I1%)
5220 POKEJ%,2048%
5230 GOSUB5990
5240 GOSUB5990
5250 IF PEEK(&0320)\256% = 0% THEN
RETURN
```

```
5260 I2% = I2% + 1%
5270 IF Z1% = 1% GOTO 5340
5280 PRINT" СЧИТЫВАНИЯ #";I2%
5290 IF I2%<3% GOTO 5010
5300 PRINT"ПРОЧИСТИТЕ ГОЛОВКУ, ЗАТЕМ НАЖМИТЕ КЛ. ВВОД";
5310 GOSUB6080
5320 INPUT I1%
5330 IF Z1% = 0% GOTO 5002 ELSE RETURN
5340 PRINT "КОНТРОЛЬНОГО СЧИТЫВАНИЯ"
5350 GOTO 5300
5360 POKE&0324,J2%
5370 POKE&0320,2%
5380 I1% = 050%
5390 GOSUB6000
5400 I1% = USR2(I1%)
5410 POKEJ%,2048%
5420 GOSUB5990
5430 GOSUB5990
5440 Z1% = 1%
5442 R2% = R1%
5445 R1% = 050%
5450 Z% = 0%
5460 GOSUB5010
5465 R1% = R2%
5470 Z% = 1%
5480 IF I2%><0% GOTO 5002 ELSE RETURN

5990 I1% = 4%
6000 FOR I% = 0%TOI1%
6010 I3% = SIN(I%)
6020 NEXT I%
6030 IF I1% = 4% THEN POKEJ%,0%
6040 RETURN

6080 FOR I% = 0%TO40%
6090 BEEP
6100 NEXT I%
6110 RETURN

    9100 FOR I% = T1%TO2% STEP2%
    9110 PRINT I%,PEEK(I%)
    9120 NEXT I%
    9130 RETURN

    280 DIM AM%(N)
/по количеству предполагаемых записей\
    290 DATA 0000%,0000%, ....
,0000%,0000%
    300 FOR I1% = 1%TO N
    310 READ I%
    320 AM%(I1%) = I%
    330 NEXT I1%
```

Здесь N — количество предполагаемых записей.

Перед описанием программы необходимо напомнить, что порт ввода-вывода компьютера БК-0010 имеет адрес ячейки памяти &0177714 как для ввода, так и для вывода.

Как уже было упомянуто, программа фактически состоит из трех подпрограмм. Текст подпрограммы в машинных кодах содержится в операторе 210. Ее начальный адрес и номер задаются в операторе 260. Назначение этой подпрограммы — пересылка в необходимое место ОЗУ имени загружаемого или считываемого с магнитной ленты файла. Подпрограмма записи или считывания данных в двоичном коде уже имеется в ПЗУ компьютера. В операторе 270 задается ее стартовый адрес. Более подробно с работой этих подпрограмм можно ознакомиться в названной выше статье (№ 9, 1988 г.).

Главная часть программы содержится в операторах 5000-6110. Сюда входят и две служебные подпрограммы. Подпрограмма 6000-6040 — управляемая выдержка времени, если обратиться к этой подпрограмме с адреса 5990, то выполняется еще и очистка выходного порта. Подпрограмма 6080-6110 — звуковой сигнал.

Управление режимами работы магнитофона выполняется с помощью оператора POKE, с помощью которого по адресу выходного порта (&0177714) записываются определенные числа. С помощью оператора PEEK считывается состояние определенных разрядов входного порта. Разряды портов ввода/вывода и их назначение приведены в табл.1.

Вызов предлагаемой программы программами пользователя должен выполняться посредством оператора GOSUB 5000, вставляемого в соответствующие места программы пользователя. Управление программой выполняется с помощью переменных,

Таблица 1

Порт	Разряд порта	Код	Контакты "УП"	Назначение
Вывод	9	512	B28	Перемотка к концу
-"-	10	1024	A27	Перемотка к началу
-"-	11	2048	B27	Стоп
-"-	12	4096	A26	Воспроизведение
-"-	12,13	12288	A26, B26	Запись
Ввод	0	B24		Ввод сигнала с оптодатчика
-"-	1	A24		Ввод сигнала "Стоп"

значения которых должны быть присвоены до выполнения оператора GOSUB 5000. Имена переменных и их значения приведены в табл.2.

Переменная Z% задает основные режимы работы программы — считывание или запись. Переменная NF$ определяет имя файла. Имя может быть индивидуальным для каждого файла или общим для всех. Лучше, если длина этой переменной всегда будет составлять 16 символов. Переменная R1% определяет адрес файла на магнитной ленте. Так как на магнитную ленту можно записать несколько десятков файлов, то их адреса лучше свести в массив и по мере необходимости присваивать значения определенных элементов этого массива переменной R1%. Значения адресов должны быть определены до начала работы программы. Для повышения надежности записи в программе непосредственно после записи информации на ленту выполняется ее контрольное считывание и в случае неуспешного считывания операция записи повторяется, и так до 3 раз. Начало области контрольного считывания задается переменной J3%. Под контрольное считывание должен отводиться такой же по объему участок ОЗУ, как для выводимого массива. Совмещать эти области нельзя, так как при неудачном контрольном считывании будет испорчена информация в рабочем массиве, что не позволит выполнить правильную повторную запись. В программе также используются переменные: J% — восьмеричный адрес порта ввода-вывода; I2% — счетчик количества ошибочных считываний; Z1% — вспомогательная переменная, показывающая, что при Z1% = 0% выполняется считывание рабочего массива, при Z1% = 1% выполняется контрольное считывание.

Для определения переменных J1%, J2%, J3% служит вспомогательная подпрограмма 9100-9130, которая используется только при настройке программы.

Таблица 2

Имя	Значение	Функция в программе
Z%	0%	Считывание данных с магнитной ленты
	1%	Запись данных на магнитную ленту
NF$	По выбору пользователя	Имя файла данных (не более 16 символов)
R1%	0%-7000%	Адрес файла данных на магнитной ленте (определяется до начала работы программы)
J1%	Определяются перед началом работы программы	Адрес начала файла данных в ОЗУ
J2%		Длина записываемого файла
J3%		Адрес начала служебной области для контрольного считывания записанного файла

Методика определения J1%, J2%, J3% и R1% будет приведена ниже. Если пользователь считает, что контрольное считывание излишне, можно удалить из программы операторы или их части, выделенные в тексте программы. Но стоит заметить, что даже на магнитофоне сравнительно высокого класса, каким является "Нота-225", ошибки при работе с магнитной лентой не редкое явление.

Принципиальная схема аппаратной части приведена на рисунке. В схеме использовано пять корпусов микросхем наиболее распространенных серий 1 шт. К140УД8А, 1 шт. К155ЛА3, 3 шт. К155ЛА8. При необходимости микросхемы можно заменить другими, имеющими те же функции.

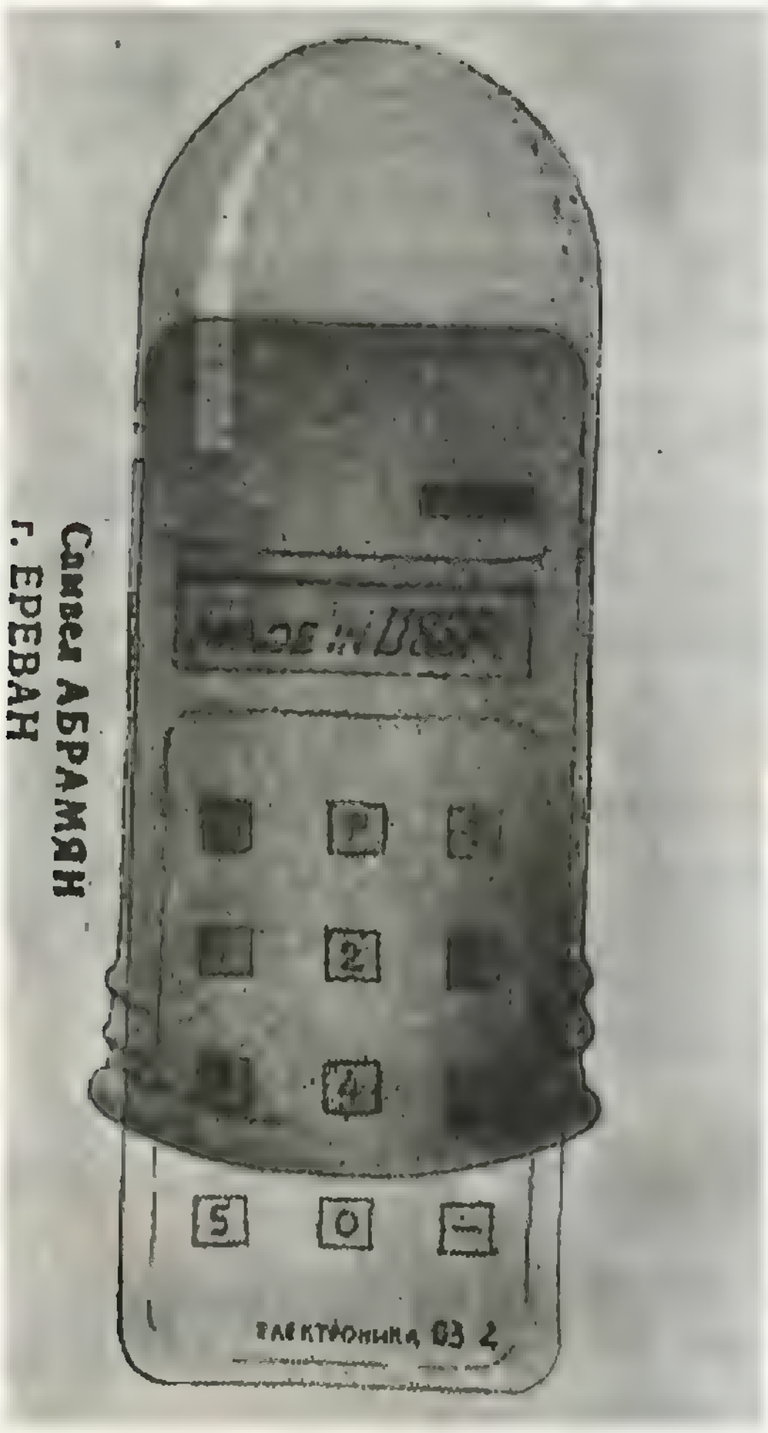
Разъемы магнитофона, к которым подключается схема, имеют обозначения и нумерацию контактов в соответствии с принципиальной схемой, приводимой в описании магнитофона "Нота-225".

Функционально схему можно разделить на две части: формирователь сигналов управления компьютера, выполненный на DD1 или -DD3, и формирователь сигналов оптодатчика и автостопа на DA1, DD4. Формирователь сигналов управления усиливает сигналы выходного порта компьютера, соответствующие выходы микросхем подключаются параллельно контактам клавиш управления магнитофона. Оптодатчик магнитофона служит датчиком расхода ленты. Формирователь сигнала оптодатчика содержит компаратор на микросхеме DA1. Сигнал автостопа используется для определения достижения начала ленты при перемотке к началу.

В качестве варианта монтажа схемы можно предложить разместить формирователи каждый на своей плате. Плата формирователя оптодатчика и автостопа встраивается в магнитофон, а плата формирователя сигналов управления компьютера устанавливается непосредственно в коробку разъема, подключаемого к выходу "УП" компьютера. Соединительный кабель между компьютером и магнитофоном может достигать 5 м.

При правильном монтаже и кондиционных деталях устройство начинает работать сразу и не нуждается в настройке. Возможно, потребуется лишь отрегулировать пороги срабатывания компаратора с помощью подстроечного резистора R10.

Опробование работы аппаратной части выполняют следующим образом. Включив компьютер и магнитофон (следует отметить, что компьютер необходимо включать первым), с помощью оператора POKE по адресу &0177714 записывают коды, приведенные в табл. 1, при этом должен включиться соответствующий режим магнитофона. Для перехода в другой режим предварительно нужно очистить выходной порт, т.е. набрать оператор POKE &0177714,0%. Следует проверить включение всех режимов магнитофона. Для проверки автопоиска и возможности автоматической записи-считывания необходимо набрать на компьютере приведенный выше текст программ, включив в него дополнительно оператор 280 STOP. После проверки и устранения всех



Проверено электроникой, обработано силиконом

ошибок следует запустить программу. Получив на экране сообщение СТОП В СТРОКЕ 280 следует набрать с клавиатуры следующие команды:

```
J1% = 12000%
J2% = 1000%
J3% = 20000%
NF$ = "имя файла
R1% = 800%
Z% = 1%
```

Затем установить на магнитофон новую кассету, желательно без записей, перемотать ее к началу и обнулить механический счетчик магнитофона. Далее следует отмотать ленту от начала на произвольное расстояние, например до середины кассеты. После этого набирается команда GOSUB 5000. Все дальнейшие действия магнитофона должны выполняться автоматически под управлением компьютера в следующем порядке:

1 — перемотка к началу;
2 — включение автостопа и перемотка к концу до адреса 100 по механическому счетчику магнитофона;
3 — запись на ленту;
4 — по окончании записи отмотка ленты к началу записанного фрагмента;
5 — включение воспроизведения, считывание только что записанной информации.

После этого программа заканчивает работу, о чем свидетельствует появление на экране символов ОК. Следует обратить внимание на то, что запись должна начаться с адреса 100 по механическому счетчику, так как между программным адресом файла на ленте R1% и адресом по механическому счетчику ленты существует жесткая связь. В случае применения магнитофона "Нота-225" эта связь выражается уравнением $R1\% = 8 \cdot N_{мех}$, где $N_{мех}$ — показания механического счетчика.

Далее можно проверить считывание информации, для этого снова запускают программу и после ее остановки задают переменным значения, которые были указаны выше, но переменной Z% присваивается значение 0%. Потом набирается команда GOSUB 5000 и магнитофон должен выполнить действия, указанные выше, за исключением пунктов 3 и 4. Можно также проверить работу системы, задавая другие значения переменным, однако следует помнить, что если значение J1% при считывании информации с ленты попадет в область расположения программы, программа будет испорчена.

Запуск программы в эксплуатацию

Перед началом работы программы необходимо выполнить некоторые предварительные действия. Эти действия выполняются один раз перед началом эксплуатации программы. В дальнейшем никаких дополнительных действий не требуется и работа системы автоматического управления накопителем на магнитной ленте (бытовым магнитофоном) в составе программы пользователя не имеет отличий от работы обычных программ.

Прежде всего необходимо включить все операторы приведенной программы в программу пользователя. При этом нужно помнить, что операторы с номерами 210—270 должны быть включены в текст до начала выполнения основных или циклических действий, лучше их включить сразу после описания массивов программы пользователя. Операторы с номерами 5000—9130 лучше включить в текст после всех операторов программы пользователя. Конечно же, перед включением любых новых операторов в программу пользователя необходимо внимательно просмотреть обе программы, чтобы не было перекрытия номеров операторов или имен переменных. Если адреса файлов на ленте предполагается свести в массив, то заполнение этого массива удобно выполнить с помощью оператора DATA. В этом случае в программу необходимо дополнительно включить операторы 280—330. При записи их в программу пользователя символ N в

операторах 280 и 300 нужно заменить на цифру соответственно предполагаемому количеству записей. Первоначально все элементы оператора DATA должны быть нулями, причем каждое число содержит четыре нуля. Конкретными значениями эти нули будут заменены позже.

В те места программы пользователя, где необходим вывод на ленту или ввод с ленты, вставляется оператор GOSUB 5000 с соответствующим номером. Перед этим оператором необходимо задать значения переменных, управляющих работой подпрограммы: Z%, NF$, R1%, J1%, J2%, J3%. Значения Z% и NF$ можно задать сразу, а значения R1%, J1%, J2%, J3%, которые к этому моменту еще неизвестны, можно задавать следующим образом. Если R1% задает непосредственно адрес, то включается оператор R1% = 0000%, а позже, после определения значений адресов на ленте, нули заменяются конкретными цифрами. Если же адреса сведены в массив, то включается оператор R1% = AM%(00%) и индекс массива вписывается также после определения значений адресов. Если выполняется запись на ленту одних и тех же массивов и область контрольного считывания не меняется, как бывает в большинстве случаев, то в программу до выполнения основных действий можно непосредственно после операторов 210—330 включить операторы

```
J1% = 0000%
J2% = 0000%
J3% = 0000%
```

с соответствующими номерами. Конкретные значения вместо нулей будут вписаны после определения адресов области памяти, "сбрасываемой" на ленту, и области контрольного считывания.

После того как в программу пользователя включены все необходимые операторы, нужно выполнить следующие действия:

1) определение участка ОЗУ, "сбрасываемого" на ленту;

2) определение области контрольного считывания;

3) определение адресов записей на ленте и заполнение массива адресов.

1. Следует отметить, что на ленту может "сбрасываться" как один массив, так и сразу несколько, если они располагаются в памяти последовательно. Массивы могут быть как одно-, так и двумерными. Считывать можно столько массивов, сколько было записано, т.е. если было записано несколько массивов, то считать один, взятый отдельно, невозможно. Если на ленту будет записываться несколько массивов, то они должны описываться в операторе DIM непосредственно друг за другом.

Предлагаемая программа, собственно, не различает, что "сбрасывается" на ленту — один массив или несколько. На ленту записывается выбранный участок ОЗУ, определяемый начальным адресом J1% и длиной J2%. Для вывода определенного массива необходимо найти адреса его первого и последнего элементов. Разница между ними даст длину массива. Так как в версии БЕЙСИКа БК отсутствует функция VARPTR, позволяющая определить физический адрес переменной, то адреса необходимых ячеек памяти нужно выполнять "вручную". Для этой цели предназначена подпрограмма, начинающаяся с оператора 9100. Фактически она выводит на экран номера ячеек памяти и их содержимое с адреса, определяемого переменной T1%, по адрес, определяемый переменной T2%. Для определения необходимых адресов надо запустить программу и после начала ее работы остановить ее нажатием на клавишу "Стоп" компьютера.

Следует помнить, что останавливать программу можно только после выполнения всех операторов DIM, но до начала заполнения массивов конкретной информацией, предназначенных для вывода. Так как вручную уловить этот момент практически невозможно, удобно останавливать программу, когда она запрашивает какие-либо данные с помощью оператора INPUT, поместив этот оператор

в нужное место программы. При правильной остановке программы память компьютера имеет законченное распределение, причем во все элементы всех массивов записаны нулевые значения. По этому признаку массивы легко отличить от другого содержимого ОЗУ. Между собой массивы различаются тем, что перед первым элементом записываются размеры массива. После нажатия клавиши "Стоп" необходимо задать адреса для просмотра содержимого ОЗУ, для этого с клавиатуры задаются значения T1% = XXXXX% и T2% = УУУУУ%.

Сначала эти адреса задаются в большой степени произвольно, с учетом того, что память под данные отводится после текста программы и обычно данные располагаются в области адресов с 8000 по 16000 (здесь и далее значения адресов приводятся в десятичной форме). Таким образом, задав начальные значения T1% и T2%, начинаем просмотр содержимого ОЗУ, для чего необходимо набрать команду GOSUB 9100. Как только на экране пойдут значения ячеек памяти, равные 0, нужно снова нажать на клавишу "Стоп". Так как вывод значений на экран происходит быстро, велика вероятность того, что место, где начинается вывод нулей, будет пропущено, в этом случае необходимо сузить интервал поиска — задать большее значение T1% и меньшее T2%. Целью поиска должно быть получение первого нулевого значения и его адреса. Адрес на экран выводится в левой колонке, а значение — в правой. После нахождения таких значений, нужно проверить тот ли это массив. Для этого необходимо набрать на клавиатуре NN%(0%) = 1%. Вместо NN нужно указать реальное имя массива в программе, а также его реальный тип. Вместо единицы можно взять любое произвольное число, главное, чтобы оно было не равно нулю. После этого необходимо вновь набрать команду GOSUB 9100 и посмотреть, изменилось ли значение ячейки памяти, где раньше находился первый из нулей. Если это значение изменилось, значит определен начальный адрес области считывания, т.е. значение J1%. Следует отметить, что важен сам факт изменения, а не число, появляющееся в ячейке памяти, так как при записи 1 в массив действительных чисел содержимое ячейки памяти не будет равно 1, но на это не стоит обращать внимания. Для определения конечного адреса нужно определить адрес ячейки, содержащей последний элемент последнего выводимого массива. В принципе поиск выполняется также, но искать необходимо последнее нулевое значение, за которым следуют ненулевые. Соответственно для проверки правильности найденной ячейки ненулевое значение следует присваивать не первому, а последнему элементу массива, т.е. имеющему максимальный индекс. Найденное значение начала считываемой области необходимо присвоить переменной J1% вместо нуля, записанного в соответствующем операторе. Значение J2% находится как разность между значениями конца и начала считываемой области. При присваивании значений следует помнить, что общее количество цифр в значении должно остаться неизменным.

2. Далее необходимо определить область контрольного считывания, если оно выполняется. Проще всего выполнять контрольное считывание в экранную область памяти, конечно, если программа позволяет портить информацию на экране при записи. При этом адрес начала области контрольного считывания необходимо выбирать в интервале 16384 — 32768. Необходимо лишь проследить, чтобы область контрольного считывания целиком входила в этот интервал, т.е. чтобы ее максимальный адрес, который явно не задается, равный начальному адресу плюс объем области (численно равный значению J2%), не превышал значения 32768. Выбранное значение адреса начала области контрольного считывания необходимо присвоить переменной J3% вместо нулевого значения.

3. Теперь можно заполнить массив адресов записей на ленте, если он используется, или просто определить адреса записей, если массив не используется. Для определения адресов необходимо выполнить одну или несколько пробных записей на ленту.

Действия, выполняемые при поиске значений адресов записей на ленте, во многом повторяют действия, которые выполнялись при проверке аппаратной части системы. Как и ранее, необходимо запустить программу и после начала ее работы остановить выполнение нажатием на клавишу "Стоп". До останова программы должны быть присвоены значения переменным J1%, J2%, J3% и NF$.

Далее с клавиатуры должно быть набрано Z% = 1% (задается режим записи информации на ленту), R 1% = NNNN% (задается адрес на ленте, по которому должна быть выполнена запись), где NNNN - программный адрес файла на ленте. При первой записи рекомендуется выбирать R1% = 100%-200%, чтобы запись была выполнена в начале кассеты. Затем необходимо набрать команду GOSUB 5000. Магнитофон выполнит запись по указанному адресу. Возможно, придется изменить значение, присваиваемое переменной R1% в операторе 5445. Это значение определяет величину отмотки ленты влево после выполнения записи. Значение, приведенное в программе, соответствует записи на ленту информации объемом около 1 Кбайта. При выводе больших или меньших объемов информации значение, стоящее в операторе 5445 справа от знака равенства, должно быть соответственно увеличено или уменьшено. При корректировке этого значения необходимо следить, чтобы после выполнения отмотки лента останавливалась перед только что записанным участком с небольшим запасом. Значение в операторе 5445 корректируется один раз и в дальнейшем остается неизменным.

Разница между показаниями счетчика магнитофона в начале записи и после ее окончания дает расход ленты на одну запись. Зная этот расход, можно определить количество записей и их адреса. Расход ленты на одну запись рекомендуется измерить на нескольких интервалах, например в начале, в середине и в конце кассеты, задав соответствующее значение R1%. Если после определения адресов для всего количества необходимых записей на ленте еще осталось место, то для этого места также можно определить адреса и использовать их в качестве резервных, т.е. если окажется невозможной запись по какому-либо адресу (например, из-за дефекта ленты), то адрес этой записи можно будет заменить на резервный.

После определения адресов всех записей на ленте необходимо записать все найденные значения в массив адресов AM%, т.е. изменить нулевые значения соответствующего оператора DATA на определенные. При замене следует помнить, что количество цифр в каждом числе должно остаться неизменным, т.е. если адрес равен, например, 250%, то записывать его в оператор DATA следует 0250%. Это необходимо потому, что изменение длины любого оператора программы ведет к изменению распределения памяти компьютера, следовательно, при этом как минимум будет меняться значение J1%. Поэтому при самых малых изменениях, вносимых в программу, необходимо заново определить значение J1%. Чтобы уменьшить количество определений значения J1%, необходимо сначала отладить основную программу, а затем уже включать в нее операторы программы, обеспечивающие автоматизацию работы магнитофона.

Описанная система была опробована в работе. Результаты пробной эксплуатации показали, что возможно создание базы данных, содержащей небольшое количество (до 10—15) файлов. При большем количестве файлов, особенно когда они располагаются на ленте ближе к концу кассеты, сказываются сбои из-за неточной

перемотки, вызываемой заеданиями ленты в кассете при интенсивной эксплуатации. Хотя программа позволяет контролировать правильность записи информации, в целом надежность автоматической записи-считывания невысокая, поэтому база данных должна содержать сведения, потеря части которых не приведет к серьезным последствиям.

Описанную систему можно применить также для автоматического поиска и воспроизведения фонограмм, что позволит формировать с помощью компьютера музыкальные каталоги и концертные программы.

Для контактов:
428035, г.Чебоксары Чувашской АССР, ул.Университетская,8,кв.228.

(Окончание. Начало на с.16)

```
70 IF KF% = 0% THEN ? "К573РФ1 (IРФ2)" ELSE
? "К573РФ2(II)"
80 LOCATE 5%,5%,0% 'Убрать курсор
100 ? "Ввод с клавиатуры"
120 ?AT(5%,7%); "Ввод из ППЗУ"
140 ?AT(5%,9%); "Вывод в ППЗУ"
160 ?AT(5%,11%); "Корректировка"
180 ?AT(5%,13%); "РФ1(IРФ2) или IIРФ2"
200 ?AT (5%,15%); "Контроль"
210 ?AT(5%,17%); "Выход"
220 Y1% = Y%
221 LINE (30%,Y%)-(200%,Y% + 20%),1%,B
'Нарисовать рамку темы меню
222 C$ = INKEY$
223 IF C$ = "" THEN 222 ELSE T% = ASC(C$)
'Между кавычками нет пробела
224 IF T% = 26% THEN IF Y%»43% THEN
Y% = Y%-20% ELSE Y% = 163%
225 IF T% = 27% THEN IF Y%«163% THEN
Y% = Y% + 20% ELSE Y% = 43%
226 IF T% = 10% THEN IF Y% = 163% THEN
STOP ELSE 230
227 IF Y%«»Y1% THEN LINE (30%,Y1%)-
(200%,Y1% + 20%),0%,B
'Стереть рамку темы меню
228 GOTO 220
230 LOCATE 1%,1%,1% 'Восстановить курсор
240 ON (Y%-23%)/20% GOSUB
6000,4000,5000,3000
250 IF Y% = 123% THEN 310 ELSE IF Y%«»143%
GOTO 40
260 CO% = 1%
270 GOSUB 4000
280 ?,, "Для возврата в меню - «ввод»" 'Обратите внимание на запятые
290 IF INKEY$ = "" THEN 290 ELSE 40 'Между кавычками нет пробела
310 IF KF% = 0% THEN SI% = 1% ELSE
SI% = 0%
320 GOTO 40
1000 T% = 1%
1010 E% = 0%
1020 W% = 0%
1030 IF LEN(A$) = 1% GOTO 1200
1040 Y$ = "" + MID$(A$,T%,1%) + " " 'В первых кавычках пробела нет
1050 U% = VAL(Y$)
1060 C% = 0%
1070 IF U%«»0% THEN C% = U%
1080 IF ASC(Y$)»70 OR U%«»0% OR Y$ = "O"
GOTO 1140
1090 C% = ASC(Y$)-55%
1140 IF T%»1% GOTO 1190
1150 W% = C%
1170 T% = 2%
1180 GOTO 1040
1190 E% = W%*16% + C%
1200 RETURN
3000 I% = KF%
3010 CLS
3020 ? "Перемещение:←,→,↑,↓; Замена: C Выход: Q"
3040 GOSUB 7000
3050 X% = 0%
3060 K$ = INKEY$
3070 IF K$ = "" THEN 3060 ELSE R% = ASC(K$)
'Между кавычками пробела нет
3080 IF FIX(SQR(R%)) = 5% THEN ON R%-24%
GOTO 3210,3170,3190
3090 IF R% = 81% THEN 3360 ELSE IF R% = 67%
THEN 3140 ELSE IF R% = 8% THEN 3270 ELSE
3060
3140 GOSUB 8000
3150 D%(I%-KF%,X%) = E%
3160 GOTO 3340
3170 I% = I%-1%
3180 IF I%«KF% THEN I% = KF% ELSE 3340
3185 GOTO 3060
3190 I% = I% + 1%
3200 IF I%»KF% + 63% THEN I% = KF% + 63%
ELSE 3340
3205 GOTO 3060
3210 J% = J% + 3%
3220 X% = X% + 1%
3230 IF J%« = 51% THEN 3260
3240 J% = 51%
```

```
3250 X% = 15%
3260 GOTO 3320
3270 J% = J%-3%
3280 X% = X%-1%
3290 IF J%» = 6% THEN 3320
3300 J% = 6%
3310 X% = 0%
3320 LOCATE J%,5%
3330 GOTO 3060
3340 ? CHR$(155%)
3350 GOTO 3010
3360 ? CHR$(155%)
3370 RETURN
4000 CLS
4005 ? "ЖДИТЕ..."
4007 K% = KF%*16% + &H8000
4010 FOR I% = 0% TO 63%
4015 L% = I%*16% + K%
4020 FOR J% = 0% TO 15%
4030 N% = L% + J%
4040 POKE &HFFCC,N%
4050 IF CO% = 1% GOTO 4100
4060 D%(I%,J%) = PEEK(&HFFCC)
4070 NEXT J%,I%
4090 RETURN
4100 IF D%(I%,J%) = PEEK(&HFFCC) GOTO 4070
4140 ? "ОШИБКА ПО АДРЕ-
СУ";HEX$(I% + KF%);HEX$(J%);"H" 'После "у"
пробел
4160 GOTO 4070
5000 ? "ЖДИТЕ..."
5005 K% = KF%*16% + 2048%
5010 FOR I% = 0% TO 63%
5015 L% = I%*16% + K%
5020 FOR J% = 0% TO 15%
5030 N% = L% + J%
5040 FOR W% = 0% TO 5%
5050 POKE &HFFCC,&H8800
5060 POKE &HFFCC,N% + &H8000
5070 PORE &HFFCC,N%
5080 POKE &HFFCC,D%(I%,J%) + 30720%
5090 FOR T = 0 TO 50
5100 NEXT T,W%,J%,I%
5110 BEEP
5120 RETURN
6000 FOR I% = KF% TO 63% + KF%
6010 CLS
6020 ? "ВЫХОД: Q"
6030 GOSUB 7000
6050 GOSUB 8000
6070 IF A$ = "Q" GOTO 3360
6075 GOSUB 1000
6077 IF LEN(A$) = 1% GOTO 6090
6080 D%(I%-KF%,(J%-6%)/3%) = E%
6090 J% = J% + 3%
6100 IF J%»51% THEN 6110 EKSE 6050
6110 ? CHR$(155%)
6130 NEXT I%
6135 J% = 6%
6140 RETURN
7000 ? AT(0%,5%);HEX$(I%);CHR$(155%)
7010 FOR J% = 0% TO 15%
7020 ? AT(J%*3% + 6%,3%);HEX$(J%) 'Номер
байта
7030 ? AT(J%*3% + 6%,5%);HEX$(D%(I%-
KF%) AND &HFF)
7040 NEXT J%
7050 J% = 6%
7060 LOCATE 6%,5%,1%
7070 RETURN
8000 LOCATE J%-1%,5%
8010 INPUT A$
8020 ? AT(J%-1%,5%); " " 'Между кавычками
пробел
8030 GOSUB 1000
8040 RETURN
```


Компьютерный салон "КРОНОС"

предлагает владельцам компьютеров "АТАРИ" серии XL/XE широкий выбор игровых и прикладных программ, литературу и другие услуги. Заявки на каталог направляйте по адресу: 200032, Таллин-32, абонентский ящик № 2061.

Одинокий гений программирования

В мире информатики и вычислительной техники все настойчивее раздаются требования признать продукцию программистов интеллектуальной собственностью, распространив на нее тот же юридический статус авторского права, что и на книги или музыкальные сочинения. С этим, пожалуй, никто не спорит. Никто, кроме Ричарда Столмана, одного из лучших американских программистов.

Вот уже более пяти лет Ричард в одиночку борется с компьютерными фирмами, отстаивая свою точку зрения. Он убежден, что программы для ЭВМ должны быть всеобщим достоянием, без всяких ограничений на право их копирования и распространения. Проводя дни и ночи в лаборатории искусственного интеллекта при Массачусетском технологическом институте, Столман разрабатывает великолепные, по всеобщему мнению, программы и рассылает их всем желающим.

Долгое время фирмы-разработчики ЭВМ создавали программы, пригодные для работы только на определенных, "своих" компьютерах, ревниво следя, чтобы машины конкурентных компаний не смогли извлечь из этого выгоду. В результате на компьютерном рынке воцарился хаос. Купив компьютер, нередко приходилось тратить немало времени на приобретение подходящих для него программ. И если покупателя не удовлетворяла та или иная программа, он попадал в трудное положение, поскольку выбора у него не было. Тип компьютера диктовал тип программного обеспечения.

Однако в конце концов компьютерные фирмы вынуждены были учесть недовольство покупателей. Образовалось даже содружество конкурировавших прежде компаний, решивших объединить усилия для создания универсальной управляющей программы. На ее базе могут работать все другие программы независимо от типа компьютера. Таким образом завеса строжайшей секретности над программами приоткрылась.

Однако Ричарду, которого называют "одиноким гением программирования", этого мало. Он хочет не только открыть всем работающим на ЭВМ путь к сокровищнице программ, созданных лучшими специалистами, но и сделать это бесплатно. "Нельзя превращать человеческие идеи и знания в средство наживы, — считает Ричард. — Это сдерживает распространение знаний. Я хочу собрать все самое ценное из опыта моих коллег и создать программное обеспечение на все случаи жизни. Никто не должен зависеть от программ, продаваемых фирмами".

Этот нелегкий труд Столман начал несколько лет назад, и сейчас его работа близка к завершению. По мнению многих специалистов, даже тех, кто отнюдь не в восторге от его альтруистского "почина", качество программы энтузиаста-одиночки не уступает продукции ведущих мировых фирм. К примеру, построенный недавно лучшим американским разработчиком ЭВМ Стивом Джорбсом знаменитый персональный компьютер "Некст", который считается шагом в завтрашний день, в значительной степени оснащен программами Ричарда Столмана. А президент фирмы "Юнипресс" Марк Кригер недавно с досадой сказал: "Мы все считали не очень серьезными его обещания поспорить с гигантами, однако он уже опередил нас..." Программы, со-

зданные Столманом, привлекают пристальное внимание специалистов. За его работой постоянно следят представители таких известных корпораций, как "Хьюлетт-Паккард", "Сони", "Интел", "Сан майкросистемз".

Ричарда почти в любое время можно застать на работе. Но поговорить с ним оказалось куда сложнее - он всегда занят. И все же долгожданная беседа состоялась.

"Сейчас я уже не один, — сообщил Ричард — нашлось еще несколько добровольцев, которые присоединились ко мне. И мы организовали фирму "Фри софтуэр фаундейшн".

Слово "фри" в английском языке имеет два значения — бесплатный и свободный. Называть продукцию фирмы бесплатной не совсем верно. "Ведь нам самим надо покупать, скажем, магнитные ленты, чтобы записывать на них программы, — объяснил щепетильно честный Сотлман. — Однако наш покупатель оплачивает только почтовые расходы и стоимость магнитного носителя, а то, что на нем записано, то есть программы, он получает бесплатно". Что касается второго значения слова "фри", то в данном случае оно безупречно точно. Программы новой компании "свободны" и в том смысле, что любой обладатель может распространять их без всяких ограничений. Если с программами Столмана работает профессионал, он по достоинству оценит возможность читать ее текст. А это и есть тот секрет, который фирмы обычно так тщательно от всех скрывают. "Не делая из этого тайны, мы тем самым как бы приглашаем всех знающих толк в программировании работать над улучшением нашей продукции, — говорит Ричард. — У нас нет секретов от коллег". И действительно, программы фирмы "Фри софтуэр фаундейшн" — единственные в США, не имеющие защиты от копирования.

Ричард, программист-мечтатель, искренне верит, что, отдавая людям результаты своего труда, он не только способствует распространению знаний, но и утверждает новую "компьютерную этику". Как знать, быть может, его идеи победят, и тогда сложная творческая работа программистов, без которой невозможно представить современный мир, получит мощное ускорение.

Ричард Столман убежден, что человеческие знания нельзя превращать в средство наживы.

Владимир Просвиряков
"Эхо планеты"

ЭЛЕКТРОННЫЙ ПЕРЕВОДЧИК

Японская компания "Сэйко-Эпсон" разработала миниатюрный компьютер для перевода с английского языка на японский. Для того чтобы электронный переводчик размером с небольшой школьный пенал "прочитал" английский текст, достаточно провести встроенный в него специальный сенсор вдоль напечатанной строчки. Величина букв при этом может быть от 2,8 до 7 мм. Перевод появляется затем на небольшом экране. Лексический запас компьютера — около 31 тыс. английских слов.

Новинка, которую изобретатели назвали "Тран-Про-1000", будет стоить в розничной продаже 32 тыс. иен (около 230 американских долларов). Руководители компании рассчитывают, что миниатюрный электронный переводчик будет пользоваться популярностью прежде всего у бизнесменов, ведущих дела с западными партнерами.

И.П.Березенцев,
программист (г.Пермь)

Еще раз о "Больших программах на БК 0010"

Думаю, что программа COM, приведенная в статье Сарычихина и Фролова "Большие программы для БК 0010" (ВТ № 3,1989) и доработанная Жариковым (ВТ № 2,1990), больше всего должна заинтересовать преподавателей, работающих с КУВТ-86, так как наиболее полно достоинства программы проявляются при работе с дисководом. Доработанная для работы с дисководом программа названа мной COM1.

1. Двоеточие используется при обращении к дисководу, поэтому в программе COM1 маркер окончания программы устанавливается клавишей "КТ" — конец текста..

2. Неудобен метод переключения адреса прерываний по EMT, предложенный Жариковым. Он требует доработки каждой Бейсик-программы, причем доработанные программы автономно уже не работают. Поэтому контроль за прерываниями я поручил программе COM1. Вызываемые программы доработки не требуют.

3. Мало пользы от программы, которую каждый раз надо вводить заново, поэтому в доработанной программе предусмотрена возможность повторного запуска набранной программы и записи ее на магнитный носитель.

После доработки программа стала короче. Работа программы подробно описана в статье Жарикова (ВТ № 2,1990). Остановлюсь на доработках.

Набранная программа может быть многократно запущена с адреса &O37422 (все адреса в статье восьмеричные). По этому адресу находится обращение к подпрограмме, в которой указатель очередного байта устанавливается на начало программы. Изменяется адрес прерывания по EMT. После этого управление передается БЕЙСИКу.

При прерывании по EMT управление передается на адрес &O37446, где проверяется наличие в стеке числа &O134016. Это адрес возврата из подпрограммы, в которой БЕЙСИК принимает очередную команду с клавиатуры в непосредственном режиме. Если такого кода в стеке нет, то прерывание EMT обрабатывается стандартным образом, иначе определяется номер EMT. Если EMT 6, то в R0 записывается очередной байт программы. Если он не равен 3 (коду клавиши "КТ"), то происходит возврат из прерывания, иначе восстанавливается вектор прерывания по EMT и управление передается БЕЙСИКу.

Программу COM1 лучше набирать в БЕЙСИКе. Команда в строке 90 заменяет команду DELETE. Она не влияет на работу программы, но после выхода из программы память компьютера окажется чистой. Возможна и обратная операция. Восстановив первоначальное значение ячейки &O2024 (если, конечно, оно известно), можно восстановить программу, уничтоженную командой NEW или DELETE. В ячейке &O2024 хранится адрес конца списка строк, он равен 15854-6*n, где n — число строк в программе.

Бейсик-программа COM1.ASC после запуска создаст программу в кодах, начинающуюся с адреса &O37400, передаст ей управление и самоуничтожится.

Оператор должен набрать свою программу и нажать "КТ". Начнет выполняться набранная программа. После того как работа по программе закончилась, ее можно записать на магнитный носитель командой BSA"[ТТ:]ИМЯ", &O37422, &O40000. Точный адрес конца программы можно посмотреть в ячейке &O37540.

Записанная программа может работать самостоятельно. Ее загружают из БЕЙСИКа с самозапуском. Загруженную программу можно запускать многократно из монитора или командой USR с адреса &O37422.

Автором разработана аналогичная программа, которая последовательно загружает и запускает Бейсик-программы по заранее составленному списку. После окончания работы с каждой Бейсик-программой или нажатия клавиши "СТОП" делается запрос: "повторить?". При утвердительном ответе вновь запускается хранящаяся в памяти программа, при отрицательном — загружается следующая.

Такие программы удобно использовать преподавателям в классе, оборудованном КУВТ-86. Преподаватель заранее составляет программу-план на урок. Ученику надо лишь запросить эту программу.

Еще одно ее преимущество — ученик не может посмотреть листинг контролирующей программы и повлиять на выставленную компьютером оценку.

Текст программы COM1.ASC.

```
10 DATA &O12701, &O37542, &O104006,
&O104016, &O110021, &O22700, 3, &O1372,
&O403, &O4767, 2, &O441, &O12767,
&O37542 + , &O102, &O12737, &O37446, &O30,
&O207, &O22766, &O134016, &O20, &O1402,
&O137, &O100112, &O11667
20 DATA &O50, &O162767, 2, &O42, &O22777,
&O104006, &O34, &O1365, &O117700, &O30,
&O5267, &O24, &O22700, 3, &O1401, 2,
&O12737, &O100112, &O30, &O137, &O120234,0
30 I% = &O37400
40 DEF USR = I%
50 READ A%
60 POKE I%,A%
70 I% = I% + 2%
80 IF A% THEN 50
90 POKE &O2024,PEEK(&O2026)
100 A = USR(A)
```

Программа COM1.

```
Адрес   Код      Ассемблер
37400   012701   MOV   #UK + 2,R1     ;ЗАПИСАТЬ В R1 АДРЕС НАЧАЛА
37402   037542                       ;ПРОГРАММЫ
37404   104006   M1:  EMT   6         ;ВВОД СИМВОЛА С КЛАВИАТУРЫ
37406   104016        EMT   16        ;ПЕЧАТЬ СИМВОЛА
37410   110021        MOVB  R0,(R1)+  ;ЗАПИСАТЬ СИМВОЛ В ТЕКСТ ПРОГРАММЫ
37412   022700        CMP   #3,R0     ;БЫЛА НАЖАТА КЛАВИША "КТ"?
37414   000003
37416   001372        BNE   M1        ;ЕСЛИ НЕТ, ТО ПРОДОЛЖИТЬ ВВОД
37420   000403        BR    M2        ;ИНАЧЕ ИНИЦИАЛИЗИРОВАТЬ ПРОГРАММУ
37422   004767   ZAG: JSR   PC,M2     ;ИНИЦИАЛИЗАЦИЯ ПРОГРАММЫ
37424   000002
37426   000441        BR    EX        ;ВЫХОД В БЕЙСИК
37430   012767   M2:  MOV   #UK + 2,UK ;ПЕРЕСЛАТЬ В УКАЗАТЕЛЬ АДРЕС
37432   037542                       ;НАЧАЛА ПРОГРАММЫ
37434   000102
37436   012737        MOV   #EMT,@#30 ;ИЗМЕНИТЬ АДРЕС ПРЕРЫВАНИЯ
37440   037446                       ;ПО EMT
37442   000030
37444   000207        RTS   PC        ;ВОЗВРАТ ИЗ ПОДПРОГРАММЫ
37446   022766   EMT: CMP   #134016,20(SP) ;НАЧАЛО ОБРАБОТКИ EMT
37450   134016
37452   000020
37454   001402        BEQ   M4        ;ЕСЛИ РАБОТА В ПРОГРАММЕ, ТО
37456   000137   M3:  JMP   @#100112  ;ПЕРЕХОД НА ОБРАБОТКУ EMT
37460   100112
37462   011667   M4:  MOV   (SP),A    ;ИНАЧЕ ОПРЕДЕЛЕНИЕ НОМЕРА EMT
37464   000050
37466   162767        SUB   #2,A
37470   000002
37472   000042
37474   022777        CMP   #104006,@A ;ЕСЛИ НЕ EMT &, ТО ПЕРЕХОД
37476   104006
```

```
37500   000034
37502   001365          BNE     M3              ;НА СТАНДАРТНУЮ ОБРАБОТКУ EMT
37504   117700          MOVB    @UK,R0          ;ИНАЧЕ ПЕРЕСЛАТЬ В R0 ОЧЕРЕДНОЙ
37506   000030                                  ;БАЙТ ПРОГРАММЫ
37510   005267          INC     UK              ;ИНКРЕМЕНТ УКАЗАТЕЛЯ
37512   000024
37514   022700          CMP     #3,R0       ;ПРОВЕРКА ОЧЕРЕДНОГО БАЙТА
37516   000003
37520   001401          BEQ     M5              ;ЕСЛИ ПРОГРАММА НЕ ОКОНЧЕНА,
37522   000002          RTI                     ;ТО ВОЗВРАТ ИЗ ПРЕРЫВАНИЯ EMT
37524   012737  M5:     MOV     #100112,@#30    ;ИНАЧЕ ВОССТАНОВЛЕНИЕ ВЕКТОРА
37526   100112                                  ;ПРЕРЫВАНИЯ ПО EMT И
37530   000030
37532   000137  EX:     JMP     @#120234        ;ВОЗВРАТ В БЕЙСИК
37534   120234
37536   000000  A:      .#0
37540   000000  UK:     .#0                     ;УКАЗАТЕЛЬ ОЧЕРЕДНОГО БАЙТА
                        END
```

Аппаратно-программные отладочные средства для ОЭВМ — грандиозные возможности сегодня, неограниченные перспективы в будущем!

Львовский научно-производственный кооператив "МОНИТОР" изготовит, поставит и внедрит:

- программные отладочные кросс-средства для однокристальных и микро-ЭВМ КР1816ВЕ48, КР1816ВЕ51, К1813ВЕ1, К580ИК80 в составе кросс-ассемблера, редактора, дизассемблера и программного эмулятора;
- программатор микросхем К573РФ2 (РФ5), КР573РФ3, КР573РФ4 (РФ6), К1816ВЕ48, К556РТ4, К556РТ5, К556РТ7, К556РТ16, К155РЕ3, INTEL 27128, 27256, 27512. Через канал "Стык-2" обеспечивается двухсторнняя связь с ПЭВМ, имеющими аналогичный интерфейс;
- аппаратные эмуляторы для отладки программ для КР1816ВЕ48 и КР1816ВЕ51;
- программные средства поддержки локальной вычислительной сети "звезда", включающей микроконтроллеры "Электроника МС2721" и центральную ПЭВМ типа ДВК;
- программные средства для ПЭВМ IBM PC;
- программное обеспечение для ПЭВМ "Львов ПК-01".

Наш адрес: 290044, г.Львов-44, а/я 8863, НПК "Монитор".

Телефоны для справок: 34-29-42, 35-35-79 с 19 часов.

В.Г.Бендаринов,
г.Горький

О программировании интерфейсов

При чтении программы, опубликованной в № 2 за 1990 г.(с. 36), управляющей программатором ППЗУ, в глаза бросилась самая начальная ее часть — выбор режима работы. Она, конечно, предельно проста и предусматривает все режимы работы, но неудобна. Возможно, пользователь БК этого сразу не поймет, но пользователю компьютеров типа IBM с более совершенным программным обеспечением этого доказывать не нужно.

Следует сформулировать основные положения, которых надо придерживаться при программировании интерфейсов:

1. Когда у компьютера возникает потребность спросить пользователя, что ему надо делать дальше, он должен представить пользователю:

а) причину останова работы;

б) меню режимов работы;

2. Когда спрашивает компьютер, именно он является инициатором, поэтому он должен ограничить действия пользователя, иначе пользователь может быть не понят, если введет команду не вовремя.

3. Компьютер безмозглый. Поэтому вследствие неаккуратности программиста может сложиться такая ситуация: когда, не навредить работе, лучше ее совсем прекратить, выключив питание компьютера (например, безальтернативный вопрос об удалении файла). Поэтому из любого режима следует предусматривать выход.

4. При составлении сервисных программ типа HELP, меню лучше, если они будут занимать меньшую часть экрана, т.е. оставлять предыдущую картинку.

5. Пользователь должен иметь полное представление о своих возможностях, т.е. иметь хорошую полную систему меню.

6. Если пользователь вводит команду с клавиатуры текстом, он может ошибиться. Поэтому в таких случаях, следует обязательно предусматривать обработку ошибок, расширенные диагностические сообщения и помощь.

7. Любое сообщение компьютера должно быть предельно ясным и не допускать разночтений.

Теперь приведем структуру таких программ в PASCAL-TURBO.

```
Program MenuExample;
Var
Koordinate: Integer;
Key : Char ;
{$I Menu . HEL}
{$I OffOld . HEL} { в эти файлы надо записать
процедуры, выполняющие }
{$I ChangeK.HEL} {соответствующие действия
и использующие глобальные}
{$I OnNew .HEL} {переменные Key и
Koordinate }
{$I Makes . HEL}
Begin
Menu; {вывод текста меню }
Repeat;
If KeyPressed Then
Begin
Key: = ReadKey;{в более ранних версиях ис-
пользовать Read(Kbd, Key)}
OffOld; {убрать отметку предыдущего режи-
ма }
ChangeKoordinates; {изменить координаты }
OnNew; {отметить новый режим}
If Key = #27 Then Exit; {выход по нажатию
ESC}
End;
Until Key = #13; {до нажатия ENTER }
Case Koordinate Of }
K1: Make1; {действие 1}
K2: Make2; {действие 2}
{включить все действия }
End;
End.
```

Заключение рецензента

Вопросы создания приемлемого как для пользователя, так и для профессионала-программиста варианта взаимодействия с программой (интерфейса) очень важны и достаточно сложны. Недаром вся история развития языков программирования и технических средств вычислительной

техники является историей развития интерфейса, развивающегося от примитивного ввода данных с помощью тумблеров и считывания информации с лампочек пульта первых ЭВМ до голосового и визуального общения с ЭВМ в компьютерах будущего пятого поколения.

В настоящее время развитие вычислительной техники привело нас к наличию широкого спектра ЭВМ, обладающих различными возможностями по организации удобного интерфейса с пользователем и программистом, начиная от программируемых микрокалькуляторов и кончая современными супер-ЭВМ производительностью миллиарды операций с плавающей точкой в секунду.

Организация взаимодействия с ЭВМ существенно зависит от того, какие технические средства ввода/вывода и программное обеспечение имеются у нее, и от того, с какой целью этот интерфейс создается: с целью ли взаимодействия с профессионалом-программистом или с целью облегчения работы неподготовленного пользователя. Во всех случаях разработка приемлемого интерфейса сопровождается многими компромиссами, связанными с большим числом факторов. К числу таких факторов относятся:

- диалоговый или расчетный характер программы;
- ограниченное время разработки и отладки;
- круг пользователей программы;
- время "активной" жизни программы;
- ограничения, накладываемые наличными ресурсами ЭВМ;
- необходимость работы программы в реальном времени;
- обеспечение надежной обратной связи с программой и многие другие влияющие факторы.

Лучшие диалоговые программные средства включают в себя десятки вложенных меню, контекстно зависимые подсказки в любом из режимов, возможность возврата назад при неверных действиях, наличие средств для создания макросов и макрокоманд для часто повторяющихся действий, графический интерфейс с использованием "мыши" или джойстика. Всего не перечислить. При этом все возможности ввода/вывода ЭВМ могут быть задействованы для диалогового интерфейса с пользователем программы или комплекса.

Практические ограничения интерфейса часто связаны только с ограниченными фантазией и умением разработчиков программ.

Положения по проектированию интерфейса, предложенные В.Г.Бендариновым, известны опытным пользователям и программистам. Они касаются разработки диалоговых программ. Надо напомнить, что игровые программы являются сугубо диалоговыми и к ним применимы эти принципы. Несмотря на общую правильность постулатов, приведенных автором, хочу остановиться на нескольких моментах.

Во-первых, из всех предложенных правил имеются исключения (особенно в игровых программах).

Во-вторых, никакие советы не заменят собственной вдумчивой проработки взаимодействия программы с пользователем. Стандартные решения приедаются.

В-третьих, каждый программист учится на собственных ошибках, так как они лучше запоминаются. Но все-таки наибольшее количество ошибок содержится в чужих программных продуктах, и все их вам не совершить.

В-четвертых, создателю диалоговых программ необходимо осваивать инженерную психологию, чтобы информации на экране было ровно столько, сколько нужно в данный момент диалога. Правильно выделить информацию и расцветить экран тоже поможет инженерная психология. Прочитайте книгу Б.Шнейдермана "Психология программирования", и пользователи ваших программ будут вам благодарны.

В-пятых, вопросы интерфейса с программами настолько важны, что разработчики систем программирования иногда включают средства со-

здания и поддержки меню в состав системных библиотек. Например, система Турбо-Паскаль для персональных ЭВМ типа IBM PC имеет специальный пакет библиотечных файлов Turbo Professional, среди которых есть эти средства, поддержанные специальной диалоговой системой. Они достаточно просты в освоении и позволяют создавать удобный интерфейс действительно высокого уровня.

Скелет программы на Турбо-Паскале, представленный автором, пригоден для простого меню, в котором тема меню выбирается только один раз. Для многократного использования меню необходимо цикл REPEAT и оператор CASE охватить еще одним циклом.

В головной программе не учтено, что управляющие символы, в частности символы, вырабатываемые клавишами перемещения курсора, требуют двух операторов ReadKey (или Read (Kbd, Key), поэтому во включаемых процедурах необходим свой анализ значения переменной Key и считывание второй половины кода для управляющей клавиши.

Кроме этого, в операторе CASE вместо К1 и К2 требуются некоторые целые значения, чтобы программа соответствовала синтаксису ПАСКАЛЯ.

Без процедур, выполняющих рисование, отметку, смену координат в меню, текст данной программы несет чисто символическую нагрузку и особой ценности без них не имеет.

Как и с помощью каких средств проектируют меню профессионалы, программирующие на БЕЙСИКЕе, можно узнать в выпуске журнала "Мир ПК", № 1, 1990 (Джим Вудграфф. Генератор меню). Статья Д.Вудграффа будет полезна не только профессионалам.

БК-ЗА-РОГА

Мелентьев А.В.
(г. Огре Латвийской ССР)

1. Многие программы гасят служебную строку, но ее место никем не используется. А если выполнить оператор POKE 136, 16384, то область служебной строки станет доступна для графики с координатами Y от 240 до 255. Если, кроме этого, сделать POKE — 76, 512 OR (PEEK (132)/64 — 40), то освобожденная область служебной строки установится снизу, т.е. получается графический экран размером 256x256 точек. Кстати, оператор CLS теперь очистит экран вместе со служебной строкой.

2. При работе БЕЙСИКа, зашитого у БК в ПЗУ, с символьными переменными возникают ошибки: полная утрата значения переменной, замена значения на совершенно абсурдное, неизвестно откуда взявшееся. Работая с БЕЙСИКом, я заметил, что во всех операторах, которые работают с константами, заключенными в кавычки, таких ошибок не возникает. Теперь я во всех операторах вставляю сложение с пустой строчкой. Например:

```
230 A$ = " " + MIO$ (B$,3,1)
240 C$ = " " + A$ + B$
```

3. В некоторых программах требуется вывод в последнюю позицию экрана, но при этом весь экран сдвигается вверх. Чтобы этого не произошло, перед выводом выполните оператор POKE 114,0. После того как символ введен, рекомендую выполнить POKE 114,2 и вернуть тем самым систему в исходное состояние.

4. В любой динамической игре важна скорость вывода на экран отдельных участков, в которых изображены герои игры. В некоторых версиях БЕЙСИКа для этого используют спрайты. Что-то подобное можно сделать и на БК. Для этого в программу следует вставить строки:

```
10 POKE 256, 7489
20 POKE 258, 2
30 POKE 260, 2527
40 POKE 262,  31244
50 POKE 264, 135
60 DEF  USR = 256
```

Каждый спрайт задается символьной последовательностью из 10 символов, код каждого из которых есть двоичная зашифровка соответствующей строки спрайта. Например, для спрайта:

1	2	4	8	16	32	64	128	
		x	x	x	x			4 + 8 + 16 + 32 = 60
		x			x			4 + 32 = 36
			x	x				8 + 16 = 24
x	x	x	x	x	x	x	x	1 + 2 + ... + 128 = 255
x		x			x		x	1 + 4 + 32 + 128 = 165
x		x			x		x	1 + 4 + 32 + 128 = 165
x		x	x	x	x		x	165 + 8 + 16 = 189
		x			x			36
		x			x			36
	x	x			x	x		36 + 2 + 64 = 102

Получаем: A$ = "<$" + CHR$ (24) + "Ъ ⊢⊢⊢ $$F". Для того чтобы ввести такой спрайт на экран, надо выполнить операторы:

```
1000 LOCATE X,Y,0
1010 A$ = USR (A$)
```

Внимание! Вывод спрайта не приводит к передвижению курсора, поэтому "увидеть" спрайт в диалоговом режиме невозможно.

5. Оператор COLOR БЕЙСИКа БК0010.01 позволяет менять цвет текста, однако локальное изменение цвета фона невозможно: весь экран меняет цвет. Что делать?

В системной области есть ячейка с адресом 138, которая содержит маску фона.

Цвет	Маска
1	-1
2	-21846
3	21845
4	0

Если надо, например, установить синий фон, выполните оператор POKE 138, 21845.

КОММЕНТАРИЙ РЕЦЕНЗЕНТА

На способ использования служебной строки в БЕЙСИКе БК0010.01

Многие программы используют решение информации в служебной строке.

Можно, например, воспользоваться советом П.Лучка из г. Кировска: удерживая клавиши АР2 и ↓, нажать клавиши 4 5 5, строка исчезнет, но вместе с курсором. Не отпуская первые две клавиши, нажимаем 1 , и курсор появляется. Вернуть обратно служебную строку можно командой NEW (журнал "Информатика и образование". — 1988. — № 6).

Вывести информацию на служебную строку в БЕЙСИКе можно также с помощью простого способа, предложенного С.Комаровым из г.Львова (журнал "Информатика и образование". — 1989. — № 4).

Способ, предложенный А.В.Мелентьевым, отличается от вышеупомянутых лаконичностью и работает как в диалоговом, так и в программном режиме. Освобожденную строку можно использовать для вывода графической информации.

На способ устранения ошибок при работе с символьными функциями БЕЙСИКа БК0010.01

Многие пользователи БЕЙСИКа БК 0010.01 с огорчением обнаруживают ошибки БЕЙСИК-системы при работе с символьными переменными. Особенно часто сбои возникают при использовании функции MID$. А.В.Мелентьев заметил, что при добавлении результата вычислений символьных функций к пустой строке ошибки не возникают, и предлагает этот способ другим пользователям БК.

Опыт использования символьных функций в БЕЙСИКе БК 0010.01 позволяет с уверенностью утверждать, что подавляющее большинство ошибок применения символьных функций связано с неверной работой функции MID$.

Надо отметить, что эта функция особая и требует к себе повышенного внимания. Это единственная функция, которая может присутствовать по обе стороны оператора присваивания, и именно с этой особенностью связаны сложности и ошибки ее реализации.

Верность ее вычисления, как показывает практика программирования, в правой части оператора присваивания обеспечена только при присвоении результата пустой символьной переменной. Например:

```
10 B$ = "ПУСТА"
20 A$ = ""      ' Очистка символьной переменной
30 A$ = MID$(B$,2%,4%)
40 ? A$;
```

Правильной работы этой функции можно достичь и способом, предлагаемым А.В.Мелентьевым, т.е. добавляя результат вычисления функции MID$ к пустой строке, например:

```
10 DIM A$(2)
20 FOR 1% = 1% TO 2%
30 A$(1%) = "KK"
40 B$ = "YY"  'Если B$ = ""ошибки не возникает
50 B$ = "" + MID$(A$(1%),2%,1%)
'Пустая строка добавляется обязательно спереди
60 ? B$;
70 ? A$(1%);
80 NEXT 1%
90 END
```

Достаточно убрать в операторе 50 пустую строку, и возникнет ошибка. В этом случае функция MID$ делает довольно странный гибрид из строк B$ и A$(2%). Меняя числа в операторе 50 и строки A$(1%) и B$, можно достичь удивительных, интересных и просто нужных результатов. Эти способы устранения ошибок при работе с символьными переменными очень полезны, так как с ними встречается практически каждый любитель, программирующий на БЕЙСИКе БК0010.01.

На способ вывода информации в последнюю позицию экрана

Существуют программы, в которых необходимо вывести символ в последнюю позицию экрана без сдвига всей информации вверх (скроллинг). Способ, предлагаемый автором, очень прост и достоин внимания.

В качестве примера использования способа предлагаю следующую программу:

```
10 POKE 114%,0%  'Отмена скроллинга
20?AT (31,23) "*";
30 C$ = ""
40 C$ = INKEY$
50 IF C$ = "" THEN 40 'Задержка до нажатия любой клавиши
60 POKE 114%,2%' Восстановление скроллинга
70 END
```

На способ создания в БЕЙСИКе БК 0010.01 спрайтов

Создание быстро выводимых изображений — один из очень важных и интересных вопросов при использовании персональных компьютеров и, в частности, бытовых ЭВМ. Существует большое количество способов кодирования и вывода на экран дисплея изображений. Предложенная автором программа достаточно лаконична для того, чтобы ее можно было включать в любую программу на БЕЙСИКе. По сути, предлагаемая программа в кодах вызывает системную подпрограмму, расположенную в адресах 102746_8-103220_8, но не с самого начала, а с адреса 102764_8, чтобы исключить некоторые начальные установки. Эта системная подпрограмма обеспечивает вывод спрайта в место расположения курсора. Возможности программы несколько ограничены из-за того, что изображение, выводимое за один вызов, имеет строго определенный размер: 8x10 точек. Большое изображение необходимо поэтому строить из многих спрайтов.

Кроме того, выводимый на экран спрайт естественным образом затирает информацию, представленную в соответствующем участке экрана. Восстановить стертую информацию может только программист.

В целом предложенный способ вывода графической информации достаточно быстрый. Например, заполнение одним и тем же спрайтом всего экрана в цикле занимает около 4-5 секунд, что соответствует приблизительно скорости вывода текстовой информации.

В примере программы, выводящей спрайт на экран, необходимо добавить дополнительный оператор LOCATE для смещения курсора в следующую позицию экрана. При его отсутствии можно потерять много времени на поиск несуществующей ошибки, так как спрайта не видно. Он будет стерт радостным сообщением БЕЙСИКа "Ок".

Предлагаю несколько иной текст тестовой программы:

```
1  REM «СПРАЙТ» БК0010.01
2  REM Автор Мелентьев А.В.
10 POKE 256%, 7489%
20 POKE 258%, 2%
30 POKE 260%, 2527%
40 POKE 262%, -31244%
50 POKE 264%, 135%
60 DEF USR = 256%
70 LOCATE 10%, 10%, 0% 'Установление позиции и гашение курсора
80  A$ = "<$" + CHR$(24) + "Ъ  Ь Ь Ь $$f"
90 A$ = USR(A$) 'Вывод спрайта
100 LOCATE 11%, 10%, 1% 'Смещение и включение курсора
110 END
```

С.Ю.Синягин (г.Дубна)

1. Что я почерпнул из встроенного монитора. Цвета, выводимые на экран, различаются только кодировкой. Например, 1 7 7 7 7 7 — это красный цвет, 0 5 2 5 2 5 — синий, 0 0 0 0 0 0 — черный. Так вот, кодировка текущего цвета должна находиться в ячейке с адресом 212, цвет переднего плана — в ячейке с адресом 214, а стандартная кодировка цветов с красного по черному находится в адресах 112026 — 112034.

2. В моих программах часто применяются две подпрограммы из монитора. Программа по адресу 110340 (можно 110346) запоминает содержимое регистров R0-R4 в стек, а программа по адресу 110362 извлекает их оттуда. Обращаться к ним надо командами JSR R4,@#110346 и JSR R4,@#110362.

3. В № 9/1988 серии "Вычислительная техника и ее применение" на стр.26 указывалась программа работы с магнитофоном по адресу 100602. Но если покопаться в мониторе, можно обнаружить интересную программу по адресу 100536. Она запрашивает имя файла на чтение и загружает его. В R5 надо указать адрес для чтения. После определенных манипуляций можно заставить эту программу запрашивать имя для записи файла на пленку.

4. Интересным свойством обладает последовательность команд:

```
EMT   6
BICB  #240, R0
```

В каком бы режиме вы ни работали — в латинском или русском, заглавных или строчных буквах, — при нажатии буквенной клавиши в младшем байте R0 будет код соответствующей заглавной латинской буквы.

КОММЕНТАРИЙ РЕЦЕНЗЕНТА

На то, что С.Синягин почерпнул, "покопавшись" во встроенном мониторе БК0010 и системной области ОЗУ

1. Стандартная кодировка цвета фона и цвета переднего плана действительно находится в ячейках с адресами 112026 — 112034, а кодировка текущего цвета фона и цвета переднего плана — в ячейках ОЗУ с адресами 212 и 214 соответственно.

Что будет в случае произвольного изменения содержимого ячеек 212 и 214, легко можно узнать, сняв защиту с системной области ОЗУ в режиме тестов командой "Щ" и заменив содержимое этих ячеек:

```
$ Щ
$ 212A <ффффффф>И,
<000000><цццццц> И /ВК/.
```

Подбирая коды фона <ффффффф> и переднего плана <цццццц>, даже в черно-белом режиме можно получать интересные эффекты.

2. Применяя программы из ПЗУ, программист получает возможность более рационально использовать область ОЗУ для своих программ.

Возможности любителей по использованию подпрограмм из встроенного ПЗУ достаточно широки как в режиме БЕЙСИКа БК0010.01, так и в режиме ФОКАЛа.

С.Синягин обнаружил в ПЗУ две очень часто используемые подпрограммы сохранения и извлечения содержимого регистров в программном стеке. В связи с этим хочется пожелать программистам не забывать, что эти программы уже имеются в ПЗУ по адресам 110346 и 110362.

3. С.Синягин нашел также способ использования подпрограммы по адресу 100536, позволяющий запрашивать имя файла для его чтения или записи. Этот способ можно рекомендовать к использованию.

Все найденные возможности применения ПЗУ и системной области ОЗУ расширяют диапазон применения БК.

4. О свойствах последовательности команд EMT 6 и BICB #240,R0.

Указанная в заголовке последова-

тельность команд действительно обладает интересными свойствами.

В любом режиме работы при нажатии буквенной клавиши в младшем байте R0 будет код соответствующей заглавной латинской буквы, при этом управляющие клавиши своих функций не меняют. Это очень удобно при программном анализе ответов на запросы программ.

Во-вторых, добавив к последовательности команд еще две команды

```
A:  EMT 6
    BICB #240,R0
    EMT 16
    BR  A
```

получим программу, позволяющую перемещать курсор в любую позицию экрана, вводить латинский текст, удалять и вставлять строки и символы и т.д. То есть получим квазиредактор для одного экрана, позволяющий полностью исследовать свойства указанной С.Синягиным пары команд. Выход из программы клавишей СТОП. Вот коротко некоторые возможности клавиш при работе с этой мини-программой:

2 — переход в левый верхний угол экрана, доступного для редактирования;

3 — удаление строки, на которой расположен курсор;

4 — вставка пустой строки перед строкой, на которой размещен курсор;

5 — перевод курсора на начало следующей строки;

6 — удаление символа справа от курсора;

7 — вставка пробела в место расположения курсора;

9 — перевод курсора вправо на одну позицию и т.д.

Особенности указанной С.Синягиным пары команд будут интересны пользователям БК0010, разрабатывающим игровые и другие диалоговые программы.

А.С.Лонгинов

Итоги конкурса на создание бытовых ПЭВМ

В прошлом году Государственный комитет по вычислительной технике и информатике, ЦК ВЛКСМ, Центральный совет ВОИР и ЦК профсоюза рабочих радиоэлектронной промышленности провели всесоюзный конкурс на создание перспективных моделей бытовых персональных ЭВМ. Итак, среди прочих достоинств модель должна быть высоконадежной, дешевой в изготовлении, иметь отечественную элементную базу и только серийную.

На конкурс было представлено более двадцати законченных разработок, выполненных как профессиональными конструкторами, так и энтузиастами-умельцами. Недавно были подведены итоги. На первом месте оказалась 16-разрядная "персоналка" МК-88, разработанная Институтом кибернетики АН УССР совместно с Минским ПО вычислительной техники. На втором — сразу две машины: 8-разрядная "Вектор-06Ц" Киевского ПО "Счетмаш" и МС-1502, разработанная в Киеве на ПО "Микропроцессор" (тоже 16-разрядная). Третьей премией награждены ПЭВМ "Эльф" Московского НИИсчетмаша и "Гуру" Челябинского ПО "Электронмаш".

У каждой персоналки — свои отличительные черты и характеристики, но какие конкретно? И еще. Кто, где и сколько будет их производить? Когда и по каким ценам продавать? Вот что рассказал по "Вектору" один из разработчиков Донат Темиразов, перешедший на работу в НИИсчетмаш:

— Главное отличие "Вектора-06Ц" от, скажем, хорошо известных компьютеров "Микроша", "Львов ПК-01", БК 0010 и других, которые иногда появляются в продаже, — отображение многоцветных графических изображений. Число цветов в палитре увеличено в несколько раз, а число одновременно отображаемых цветов — в четыре раза. Разрешающая способность экрана в "Векторе" повышена в два раза. Быстродействие компьютера и объем памяти увеличены в полтора-два раза, а сам компьютер снабжен тремя музыкальными каналами и специальным каналом для шумовых

эффектов. К "Вектору" можно подключать печатающее устройство, электронный диск емкостью 256 Кбайт, накопитель на гибком магнитном диске. В Кишиневе, Минске, Астрахани, Кирове и Волжском уже освоено серийное изготовление "Вектора-06Ц". Первая партия поступила в продажу в Кишиневе еще в прошлом году. А в этом году в торговую сеть поступит примерно 20 тысяч бытовых персоналок "Вектор-06Ц". Продажная цена компьютера зависит от завода-изготовителя и находится в пределах 750 — 850 рублей. Технический уровень соответствует мировому стандарту. Недостатком можно считать использование относительно дешевого, но морально устаревшего микропроцессора КР580ВМ80А. Начата разработка новой модели компьютера "Вектор-1608Ц". Он будет иметь два процессора — 8 и 16-разрядные, значительно больший объем памяти, накопители на гибких магнитных дисках, многоцветный (более 20 тыс. цветов) графический адаптер и работать в среде стандартных для профессиональных ПЭВМ операционных систем. Компьютер совместим с ЕС-1840, ЕС-1841 и IBM/PC-XT и предназначен для пользователей, занимающихся профессиональной деятельностью по разработке прикладных программ в домашних условиях. Серийное производство новой машины планируется начать с 1992 года.

Перед нами еще один призер конкурса — хорошо скомпонованный, внешне привлекательный "Эльф". О нем мы попросили рассказать его создателя — начальника лаборатории НИИсчетмаша Михаила Дроздова.

— "Эльф" — это даже не название, — возразил он, — а девиз, под которым данная машина принимала участие в состязании. Названий у нее будет несколько, а вот шифр единый — ПК8002. В первую очередь мы создавали ее для игр. Она стала первой советской бытовой персональной ЭВМ с динамической графикой, принцип работы которой основан на методе мультипликации. Оперативная память нашей персоналки разделена на "слои". В одном слое хранится "фон", а в остальных, число которых переменно и равно числу движущихся фигур, — изображение этих фигур. Формирование полной картинки осуществляется специальным блоком микропроцессора. Такая организация позволяет сократить до минимума время перемещения изображения из одного места экрана в другое и работать с крупными и многоцветными фигурами. По остальным характеристикам ПК8002 близок к "Вектору-06Ц". ОН формирует на экране обычного цветного телевизора шестнадцатицветное изображение из палитры в 256 цветов, имеет оперативную память емкостью 64 Кбайта и трехканальный генератор звука с программно-управляемой громкостью. Он содержит также генератор шума, создающий звуковые эффекты типа взрывов, выстрелов, шума прибоя, погони... Серийное производство машины планируется с 1991 года в Оренбурге, Тамбове, Кимовске и Вологде. Предполагается выпустить около 10 тысяч таких компьютеров с ростом выпуска в последующие годы до 100 тысяч в год. А в этом году на заводах выйдут только первые партии по нескольку десятков штук на каждом предприятии. И я думаю, что наш ПК8002 можно будет купить уже в будущем году. Ориентировочная цена 900 — 980 рублей. По техническому уровню ПК НИИсчетмаша не уступает лучшим зарубежным бытовым ЭВМ своего класса, таким, как "Коммодор 64", "Ямаха MSX", "Атари"...

Однако в нашем компьютере по сравнению с лучшими зарубежными моделями в три-четыре раза больше используемых элементов, что, конечно, увеличивает трудоемкость изготовления и стоимость. Сеййчас ведем работы по переводу большей части схемы нашего ПК на БИСы, что поможет снизить его стоимость в полтора-два раза.

П 53 **Пользователям** бытового компьютера. — М.: Знание, 1990. — 48 с. — (Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение"; № 8).

ISBN 5-07-001510-9

20 к.

В выпуске собраны наиболее интересные предложения наших читателей, описывающих свой опыт работы на бытовом компьютере БК0010.

Брошюра рассчитана на широкий круг читателей.

2404040000 **ББК 32.97**

ТЕМА *СЛЕДУЮЩЕГО* НОМЕРА:

РАДИО-ЭЛЕКТРОНИКА И СВЯЗЬ

Коды и кодирование

***ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА* И ЕЁ ПРИМЕНЕНИЕ**

Защита информации

МАТЕМАТИКА КИБЕРНЕТИКА

Математика и спорт

Научно-популярное издание

ПОЛЬЗОВАТЕЛЯМ БЫТОВОГО КОМПЬЮТЕРА

Гл.отраслевой редактор *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл.редактор *Н. А. Васильева*
Художник *В. Н. Конюхов*
Худож.редактор *И. А. Емельянова*
Техн.редактор *Т. В. Луговская*
Корректор *С. П. Ткаченко*

ИБ № 11192

Подписано к печати 05.07.90. Т 00635. Формат бумаги 70x100 1/16. Бумага офсетная. Печать офсетная. Усл.печ.л. 3,90. Усл.кр.-отт. 8,45. Уч.-изд.л. 4,30. Тираж 72 347 экз. Заказ 1180. Цена 20 коп. Издательство "Знание". 101835, ГСП, Москва, Центр, проезд Серова, д.4. Индекс заказа 904708. Отпечатано с оригинал-макета издательства "Знание" на ордена Трудового Красного Знамени Калининском полиграфическом комбинате Государственного комитета СССР по печати. 170024, г.Калинин пр. Ленина, 5.

Цена 20 коп.

Индекс 70195

Адрес подписчика:

Сол. 5-27

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

”. . .биология изучает систему, уже готовую и функционирующую, а для разработчика программных средств компьютер представляет собой как бы сосуд с атомами, которые еще следует объединить и организовать”

Алан Кэй

– один из главных разработчиков механизма разделения экрана дисплея на участки. Им разработан также манипулятор ”мышь”.

Наш адрес:
СССР,
Москва,
Центр,
проезд
Серова, 4